

Transportation Science

Publication details, including instructions for authors and subscription information:
<http://pubsonline.informs.org>

Just-In-Time Vehicle Routing for In-House Part Feeding to Assembly Lines

Simon Emde, Michael Schneider

To cite this article:

Simon Emde, Michael Schneider (2018) Just-In-Time Vehicle Routing for In-House Part Feeding to Assembly Lines. *Transportation Science* 52(3):657-672. <https://doi.org/10.1287/trsc.2018.0824>

Full terms and conditions of use: <https://pubsonline.informs.org/Publications/Librarians-Portal/PubsOnLine-Terms-and-Conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2018, INFORMS

Please scroll down for article—it is on subsequent pages



With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.

For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

Just-In-Time Vehicle Routing for In-House Part Feeding to Assembly Lines

Simon Emde,^a Michael Schneider^b

^a Technische Universität Darmstadt, 64289 Darmstadt, Germany; ^b RWTH Aachen University, 52072 Aachen, Germany

Contact: emde@bwl.tu-darmstadt.de,  <http://orcid.org/0000-0001-6932-0166> (SE); schneider@dpo.rwth-aachen.de (MS)

Received: March 3, 2017

Revised: August 31, 2017

Accepted: January 2, 2018

Published Online in Articles in Advance:

April 2, 2018

<https://doi.org/10.1287/trsc.2018.0824>

Copyright: © 2018 INFORMS

Abstract. This paper deals with the problem of routing in-house transport vehicles that feed parts to workstations in assembly plants or workshops just in time. The capacitated vehicles, typically so-called tow trains, perform their assigned route cyclically without break and provide each station with the exact quantity of parts required until the next arrival of the vehicle. Hence, the demand of each station depends on the duration of the route serving the station: The longer the route duration, the less frequently the station is visited and the higher its demand. The goal is to minimize first the number of vehicles and second the total route duration, while respecting given minimum service frequencies at the stations. We provide a mathematical formulation of this novel problem and address it by means of a large neighborhood search. The algorithm is able to solve realistic instances in acceptable time and vastly outperforms a default solver. We discuss two variants of the problem, one in which split deliveries to stations are allowed and another assuming that all stations lie on a straight line. Finally, we investigate the extent to which assuming constant demand rates may lead to problems during the day-to-day operations of the part-feeding system, where demands are not necessarily constant.

Keywords: [in-house logistics](#) • [vehicle routing](#) • [large neighborhood search](#) • [tow trains](#)

1. Introduction

Modern production systems are capable of sustaining very high production rates while at the same time offering customers a wide selection of individual product variants (so-called mass customization; see Da Silveira, Borenstein, and Fogliatto 2001). Mass customization is beneficial from a sales perspective, but it puts a lot of strain on the logistics processes because a broad range and high volume of parts and subassemblies need to be delivered to workers and machines on time. Because of the very limited space on the shop floor and high holding costs, just-in-time strategies have essentially become inevitable in many industries, like, for example, the automotive industry (Boysen et al. 2015). In a just-in-time environment, parts are only moved from one stage to the next if and when they are required.

To enable frequent small-lot deliveries to the workstations on the shop floor, many manufacturing plants use small in-house transport vehicles like tow trains. A tow train consists of a small electric vehicle towing a handful of wagons that carry parts for the productive units within the factory. The tow trains are loaded at an in-plant logistics area (commonly called a “supermarket” in just-in-time parlance). From there, they set off on predetermined routes to drop off parts at the workstations along their path. The parts are often packed in standardized bins (see, e.g., De Souza, de Carvalho, and Brizon 2008). Once finished with a tour,

they return to the supermarket to be refilled and immediately set off again on the next tour. To enhance reliability, ease communication of schedules, and ensure smooth operation, routes, once assigned, are usually not changed unless the production process is significantly altered (Vaidyanathan et al. 1999). Tow trains thus cyclically repeat their assigned milk runs (round trips).

Such just-in-time part-feeding systems are very common in high-volume flow-line production like automotive final assembly (e.g., Battini, Boysen, and Emde 2013). Similar systems have also been observed, for example, in factories producing exhaust systems (Vaidyanathan et al. 1999), diesel injectors (Akilloğlu et al. 2006), or components for farming equipment (Lian and Van Landeghem 2007).

This paper deals with the problem of routing vehicles with limited capacity in a just-in-time environment. The vehicles perform their assigned route cyclically without break and provide each workstation (station, customer) with the exact quantity required until the next arrival of the vehicle. Hence, the demand of each station depends on the duration of the route serving the station: The longer the route duration, the less frequently the station is visited and the higher its demand.

To our knowledge, the only previous publication dealing specifically with just-in-time vehicle routing is that by Vaidyanathan et al. (1999). The authors describe

an industry problem they encountered at an exhaust system plant and model it as a continuous vehicle routing problem (VRP) called the just-in-time capacitated VRP (JITCVRP). However, their model does not prevent individual stations from being visited rarely and consequently receiving large-lot deliveries that might exceed the available shelf space. In our experience from automotive original equipment manufacturers (OEMs), this can be problematic because space on the shop floor is notoriously scarce (e.g., Boysen, Fliedner, and Scholl 2009). The objective of their model is to minimize the total trip time for all used vehicles. In a production logistics environment, trip times per se are not necessarily of primary concern because distances are short and the small electric vehicles do not consume a lot of energy. From a practical perspective, minimizing the size of the vehicle fleet to reduce expenses and shop-floor congestion is often considered more important (Golz et al. 2012).

The main contributions of this paper are the following. First, we modify and extend the JITCVRP as follows:

- We consider scarce storage space at the station locations, which is an important issue in mixed-model assembly systems (see, e.g., Boysen, Fliedner, and Scholl 2009; Limère et al. 2012; Emde and Boysen 2012b). To ensure that individual deliveries do not exceed the limited shelf space, our model allows setting a minimum service frequency for stations. It is expressed as the maximum feasible elapsed time (MFET) between two consecutive visits to the workstation.
- We follow a hierarchical objective that first minimizes the number of vehicles and second minimizes total route duration. Because holding costs at the stations are already covered by the high-frequency cyclic deliveries, which can be guaranteed by setting the MFETs accordingly, the number of vehicles remains the most important cost driver (Golz et al. 2012). Apart from creating significant one-time investment cost, these vehicles also need to be maintained and (in many cases) operated by human personnel. Furthermore, given the limited space, having a large fleet of vehicles moving through the confines of the assembly plant may cause problems with regard to congestion and traffic jams, regardless of whether automated or human-operated tow trains are used.

Second, we present a mixed-integer linear program (MILP) for the resulting extended JITCVRP (JITCVRP-ext), and we develop a powerful heuristic procedure based on large neighborhood search (LNS), which is capable of finding high-quality solutions to problem instances of realistic size within run times of only a few minutes. Third, we discuss two relevant problem variants, namely, (i) allowing split deliveries when serving the stations and (ii) assuming that all stations lie on a

straight line. Fourth, we investigate the restrictiveness of the assumptions of the JITCVRP-ext and show that routes generated on the basis of average demand rates are in fact useful in the daily operation of an assembly system, although demand is typically not perfectly constant.

Note that the JITCVRP-ext extends the classic capacitated VRP (CVRP), which is already NP-hard in the strong sense and notoriously intractable (e.g., Laporte 2007). It is therefore most probably not possible for exact solution methods to solve instances of realistic size (often consisting of several hundred customers) in acceptable time.

The remainder of this paper is structured as follows. In Section 2, we review the pertinent literature. We formally define the problem in Section 3, and propose suitable solution procedures in Section 4. In Section 5, we discuss and solve two important special cases of the JITCVRP-ext. The performance of the algorithms and the effects of some assumptions of the problem are investigated on newly generated as well as known benchmark instances from the literature in Section 6. Finally, in Section 7, we conclude the paper.

2. Literature Review

The problem of routing capacitated vehicles in a just-in-time environment is obviously related to the classic CVRP. This type of problem has been very widely investigated for decades in the scientific literature. An excellent overview is provided by Toth and Vigo (2014). Existing VRPs are typically not geared toward just-in-time deliveries: they generally assume given discrete demands and either one single planning period or multiple discrete planning periods (e.g., Christofides and Beasley 1984; Baita et al. 1998; Campbell et al. 1998). VRPs with a continuous planning horizon like in our problem are very rare; for example, Francis and Smilowitz (2006) present such a problem, but the authors presuppose given fixed schedules and service frequencies.

The only previous publication dealing specifically with just-in-time vehicle routing is that by Vaidyanathan et al. (1999). Their model is a special case of ours because they disregard storage constraints at the stations. The authors propose a nonlinear mixed-integer programming (MIP) model. They also present a linear model, which relaxes the subtour elimination constraints and the demand satisfaction constraints to gain a lower bound on the number of vehicles. Moreover, they solve the JITCVRP heuristically with a two-stage procedure: first, feasible routes are generated via a modified nearest neighbor scheme; second, these routes are improved through 3-opt local search (LS). If a route can be improved by 3-opt, an attempt is made to add stations to the route that did not previously fit. The heuristic is tested on a set of 270 instances derived

from industry data, the problem size ranging from 10 to 71 stations. Information on the CPU times is not given, nor is there any information on the optimality gap regarding the total route duration. With regard to the number of vehicles, Vaidyanathan et al. (1999, p. 1091) report that “[o]n average, the heuristic solution required 0.35 vehicles more than the lower bound.” While no more detailed information about the solution quality is given, the authors mention that the problem size, “problem size and vehicle capacity interaction,” and “vehicle capacity and demand rate interaction” each have a significant effect on the solution quality, implying that the gap to the lower bound is greater for the larger instances.

Because our problem deals with the repeated delivery of goods from a single facility to multiple customers, it bears some resemblance to continuous inventory routing problems (IRPs), referred to as cyclic inventory routing. Larson (1988) and Webb and Larson (1995) were among the first to consider such problems, with recent contributions coming from, for example, Aghezzaf, Raa, and Van Landeghem (2006); Raa and Aghezzaf (2009); and Zenker, Emde, and Boysen (2016). These models invariably look at the problem from a strategic supply chain logistics perspective, where goods have to be moved over long distances in large trucks or even cargo ships. Consequently, one of the main focuses of these IRP formulations is to minimize the transportation cost by adjusting delivery frequencies such that they are cost-effective. In a production logistics context, however, because of the short distances and light electric delivery vehicles used, transportation cost is usually considered less important than fleet cost and timeliness.

Moreover, the JITCVRP is related to (i) periodic VRPs, in which a visiting pattern for each customer and vehicle routes for each individual day within the planning horizon must be determined (for recent surveys, see Irnich, Toth, and Vigo 2014; Campbell and Wilson 2014), and (ii) VRPs with fixed routes or more general VRPs with consistency considerations (Beasley 1984; Kovacs et al. 2014).

There are also some publications about planning tow train deliveries in automotive final assembly. Short surveys on the related literature can be found in Battini, Boysen, and Emde (2013) and Boysen et al. (2015). There are a few papers that at least touch on routing or scheduling such vehicles. Choi and Lee (2002) study an integrated planning problem that includes routing, scheduling, and determining the load of the tow train. They aim to minimize the deviation of actual from predetermined delivery times and adapt the classic insertion heuristic to their problem. Golz et al. (2012) present a case study from a German motor company, solving the routing and scheduling problems via a modified variant of the classic savings heuristic.

Their objective is the minimization of the number of tow trains. An industrial case study is also analyzed by Staab et al. (2016), using simulation tools to derive insights on how routing strategies affect traffic and blockages. Emde and Boysen (2012b) propose an exact algorithm based on dynamic programming to solve both the routing and timetabling of tow trains. The authors assume that the exact (discrete) part demands are already known with certainty at the time the routes are planned (i.e., they do not use constant demand rates) and that all customers lie on a straight line. Kilic and Durmusoglu (2013) seek to find good cyclic schedules for tow trains making milk runs by developing a custom heuristic. Similarly, Fathi et al. (2016) solve the combined scheduling and loading problem by way of a particle swarm optimization procedure, while Emde and Gendreau (2017) propose a hybrid tabu search/network simplex method to solve a similar problem.

Finally, there are other papers dealing more broadly with tow trains feeding parts to assembly lines: Emde and Boysen (2012a) propose a continuous location model to site supermarkets on the shop floor, whereas Emde (2017) optimizes the replenishment of the supermarket itself.

3. Problem Description

3.1. Mathematical Formulation

We define the JITCVRP-ext as follows. Let $G = (V, A)$ be a digraph, where $V = \{0, 1, \dots, n\}$ is a vertex set, and $A = \{(i, j): i, j \in V; i \neq j\}$ is an arc set such that G is a complete graph. Vertex 0 represents the depot—usually a supermarket on the shop floor—and the remaining vertices denote stations (customers). A set of m vehicles is stationed at the depot. The vehicles are often tow trains, which may either be operated by a human driver (Golz et al. 2012) or work as fully automated guided vehicles (Emde, Fliedner, and Boysen 2012). Tow trains are generally limited to about four or five wagons each because the sharp turns and narrow driving lanes make larger transporters impractical (Emde, Fliedner, and Boysen 2012). Consequently, we assume that all vehicles have an identical given capacity of Q bins.

Each station i has a given demand rate q_i . In automotive assembly, parts delivered by tow train are often packaged in standard-size bins at the depot (Emde, Fliedner, and Boysen 2012; Golz et al. 2012). Therefore q_i may be interpreted as the average number of bins consumed per time unit (TU; e.g., minute or work cycle), although q_i (and Q) can also denote other units of measure like volume or weight. We assume that the depot is always adequately stocked to supply all demand. This is not an unrealistic assumption because missing parts can cause significant problems, and plant managers are therefore very keen on ensuring a steady

part supply. The minimum service frequency that plant managers can set for certain stations to ensure that individual deliveries do not exceed the limited shelf space are realized by means of the MFET u_i that may elapse between two visits of station i .

A travel time t_{ij} is associated with each arc (i, j) in A , denoting the time the vehicle takes to go from station i to station j . In addition, to correctly evaluate empty routes, we define $t_{00} = 0$. Moreover, each station i requires s_i units of service time (i.e., the time the tow train takes to stop there and exchange empty bins for full bins), and s_0 denotes the replenishment time of the vehicle at the depot. We assume that the production process is stable; that is, the number of bins delivered and the number of bins collected can be expected to be equal on each trip, and thus vehicle capacities will not be violated by returning empties.

Routes, once assigned, are repeated cyclically without break, except the break implied by the replenishment time s_0 . A vehicle delivers exactly the number of parts to each station that is consumed until its next arrival, in line with the just-in-time philosophy. To minimize traffic congestion on the densely packed shop floor and to facilitate organizing and communicating routes and schedules, each station must lie on exactly one route, and each route is assigned to one unique vehicle. Thus, a solution to the JITCVRP-ext is defined by a partition $\{R_1, \dots, R_m\}$ of $V \setminus \{0\}$ and a permutation π_k of R_k specifying the order in which the stations on route k are visited. Note that $R_k = \emptyset$ if vehicle k is not used. Vehicle k travels along route $\pi_k = \langle \pi_k(0), \pi_k(1), \dots, \pi_k(|R_k| + 1) \rangle$, where $\pi_k(i) \in V \setminus \{0\}$: $i \neq 0 \wedge i \neq |R_k| + 1$ is the i th station to be visited by vehicle k , and $\pi_k(0) = \pi_k(|R_k| + 1) := 0$. We say that a solution is feasible if it satisfies the following conditions:

- The vehicle capacity is not violated on any route, i.e.,

$$\left(\sum_{i=0}^{|R_k|} (t_{\pi_k(i), \pi_k(i+1)} + s_{\pi_k(i)}) \right) \cdot \left(\sum_{i \in R_k} q_i \right) \leq Q \quad \forall k = 1, \dots, m.$$

The total demand on any given route depends on the duration of that route. The longer the route duration, the higher the demand that accumulates in the meantime.

• To account for storage space constraints, the time that passes in between two visits to a station must not exceed the given limit. Because routes are repeated cyclically, this means that the duration of the route that serves station i must not be greater than u_i , or

$$\sum_{i=0}^{|R_k|} (t_{\pi_k(i), \pi_k(i+1)} + s_{\pi_k(i)}) \leq u_i \quad \forall k = 1, \dots, m.$$

As described in Section 1, we follow a hierarchical objective; that is, comparing two solutions, the one

Table 1. Parameters and Decision Variables of the MILP Model

V	Set of stations, indices $i, j \in \{0, 1, \dots, n\}$, where 0 denotes the depot
M	Big integer
Q	Vehicle capacity
t_{ij}	Travel time from station i to station j
s_i	Service time at station i (replenishment time at the depot in case of s_0)
u_i	MFET of station i
q_i	Demand rate of station i (quantity per time unit)
w_i	Decision variable: remaining load on arrival at vertex i
τ_i^{\rightarrow}	Decision variable: duration of the route serving i until the visit of vertex i
τ_i^{\leftarrow}	Decision variable: duration of the route serving i from the visit of vertex i until the return to the depot
x_{ij}	Binary decision variable: 1 if arc (i, j) is traveled; 0 otherwise

with the smaller number of vehicles m is always superior. If both solutions require the same number of vehicles, we prefer the one with lower total route duration $\sum_{k=1}^m \sum_{i=0}^{|R_k|} t_{\pi_k(i), \pi_k(i+1)}$ because this implies higher service frequencies.

Table 1 summarizes the problem parameters and defines the decision variables, which we use to present the following MILP that allows the use of default solvers

$$\text{Minimize } f(\mathbf{x}, \boldsymbol{\tau}^{\rightarrow}, \boldsymbol{\tau}^{\leftarrow}, \mathbf{w}) = M \cdot \sum_{j \in V} x_{0j} + \sum_{i \in V} \sum_{\substack{j \in V: \\ i \neq j}} x_{ij} \cdot (t_{ij} + s_i) \quad (1)$$

subject to

$$\sum_{\substack{j \in V: \\ i \neq j}} x_{ij} = 1 \quad \forall i \in V \setminus \{0\}, \quad (2)$$

$$\sum_{\substack{i \in V: \\ i \neq j}} x_{ij} = 1 \quad \forall j \in V \setminus \{0\}, \quad (3)$$

$$\tau_i^{\rightarrow} \geq t_{0i} + s_0 \quad \forall i \in V \setminus \{0\}, \quad (4)$$

$$\tau_i^{\leftarrow} \geq t_{i0} + s_i \quad \forall i \in V \setminus \{0\}, \quad (5)$$

$$\tau_j^{\rightarrow} \geq \tau_i^{\rightarrow} + (t_{ij} + s_i) \cdot x_{ij} - M \cdot (1 - x_{ij}) \quad \forall i, j \in V \setminus \{0\}, i \neq j, \quad (6)$$

$$\tau_i^{\leftarrow} \geq \tau_j^{\leftarrow} + (t_{ij} + s_i) \cdot x_{ij} - M \cdot (1 - x_{ij}) \quad \forall i, j \in V \setminus \{0\}, i \neq j, \quad (7)$$

$$\tau_i^{\rightarrow} + \tau_i^{\leftarrow} \leq u_i \quad \forall i \in V \setminus \{0\}, \quad (8)$$

$$w_j \leq w_i - (\tau_i^{\rightarrow} + \tau_i^{\leftarrow}) \cdot q_i + 2 \cdot Q \cdot (1 - x_{ij}) \quad \forall i \in V \setminus \{0\}, j \in V, i \neq j, \quad (9)$$

$$0 \leq w_i \leq Q \quad \forall i \in V, \quad (10)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V; i \neq j. \quad (11)$$

Objective function (1) minimizes first the number of vehicles, then the total duration of all routes. Constraints (2) and (3) enforce that each station is entered

and left exactly once. Constraints (4) and (5) define the forward and backward duration of the route for a customer directly following or preceding the depot. Constraints (6) and (7) do the same for consecutive customer visits. Inequalities (8) ensure that the time that passes in between two visits to a station must not exceed the given MFET. Constraints (9) and (10) guarantee that maximum vehicle capacities are respected. Finally, the domain of the variables is defined in (11).

Like all mathematical models, the above MILP relies on several assumptions and simplifications with regard to the real-world problem:

1. Station demand is expressed as a constant demand rate that does not change during the planning horizon. This is clearly a simplification of reality, where demands are necessarily discrete. However, exact demands are often simply not available at the time when routes are planned. For instance, at the main plant of a major German OEM that we visited, production schedules are disturbed so frequently that planners decide on tow train routes on the basis of the production program for the day but not based on the exact production sequence. Because level part demands are a core concept of the just-in-time philosophy (the famous *heijunka* principle; Monden 2011), demand rates are often an adequate approximation.

Note that because demands are aggregated as average rates q_i , on any given trip, the actual demand when the schedule is executed may fluctuate somewhat. To account for this, it may be expedient to add a safety margin by setting Q to a slightly lower value than the actual vehicle capacity. We will investigate this issue in more detail in the computational studies in Section 6.

2. All stations have to be visited (and parts are delivered) on each execution of the route that the station is assigned to. This implies that the part consumption is high enough at each station to warrant cyclic just-in-time deliveries. This is certainly justified in high-volume production systems. Also note that if a particular station does not have a high part demand, it may be supplied with bins containing fewer parts.

3. The JITCVRP-ext is a medium-term problem. The exact load of the vehicles will be determined at a later time by the logistics workers at the depot, often relying on kanban signals. In many cases, parts will be put into standard-size bins, either filled homogeneously or packed as so-called kits that are presorted to match the assembly sequence (Limère et al. 2012). This, however, is not part of the JITCVRP-ext, which is concerned only with the total number of bins (or volume) of parts in demand, expressed by parameter q_i , and not with the exact contents of the bins.

4. Each route is served by one tow train, and each station can only lie on one route. We will relax this assumption and discuss allowing split deliveries in Section 5.

5. For the distance matrix (t_{ij}) , the triangle inequality holds.

Note that under the assumptions outlined above, defining a MFET is sufficient to model a limited shelf capacity (measured, e.g., as the maximum number of bins) at the stations. For example, assume that some station i has a shelf capacity of 10 bins and that, on average, it consumes 0.5 bins per time unit (work cycle). Then, no more than $u_i = 10/0.5 = 20$ cycles must pass in between two successive visits of the tow train (i.e., the route of the tow train serving station i must not be longer than 20 cycles). Otherwise, more than 10 bins have to be delivered at least on some tours, exceeding available storage space.

3.2. Example of a JITCVRP-ext Solution

Consider the example data from Figure 1(a), and let the vehicle capacity be $Q = 20$.

The optimal solution for this problem consists of two routes: $\langle 0, 3, 2, 1, 0 \rangle$ with total duration $t_{0,3} + t_{3,2} + t_{2,1} + t_{1,0} + s_0 + s_3 + s_2 + s_1 = 12$ and $\langle 0, 4, 0 \rangle$ with total duration $t_{0,4} + t_{4,0} + s_0 + s_4 = 9$. The total capacity demands of the routes are $12 \cdot (q_3 + q_2 + q_1) = 18 \leq 20 = Q$ and $9 \cdot q_4 = 4.5 \leq 20 = Q$, respectively. Both routes also adhere to the MFETs. They are schematically depicted in Figure 1(b).

4. Algorithms for the JITCVRP-ext

In this section, we introduce a fast nearest neighbor-based construction heuristic (NN) and a novel LNS scheme to solve the JITCVRP-ext.

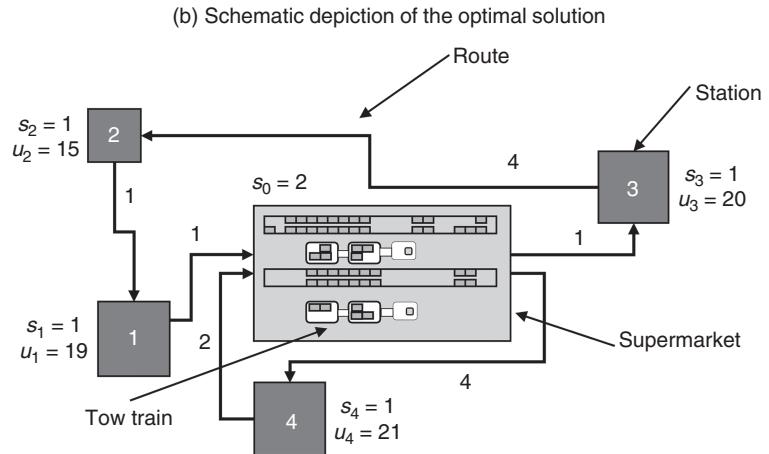
4.1. Construction Heuristic

To generate an initial feasible solution, we adapt the well-known nearest neighbor scheme (e.g., Solomon 1987) to the JITCVRP-ext as follows.

Starting from the depot, in each iteration, the unvisited station closest to the last station on the route is appended (tie breaker: lowest index), where only stations that can be added to the emerging route without violating any constraints are taken into consideration. In this step, care must be taken that lengthening a tour decreases the service frequency for all stations on the route. More specifically, let i be the last station that has been added to the current route (or the depot if no station has yet been added). Moreover, let f_i be the total route duration up to station i (s_0 if no station has yet been added), let u_i be the lowest MFET of the current route (∞ if no station has yet been added), and let q_i be the total demand rate of the route (0 if no station has yet been added). Then, a station j that has not yet been visited can feasibly be attached to the current route only if $f_i + t_{ij} + t_{j0} + s_j \leq \min\{u_i; u_j\}$ and $(f_i + t_{ij} + t_{j0} + s_j) \cdot (q_i + q_j) \leq Q$. If no such station exists, a new route is opened, beginning at the depot. If no unvisited stations are left, the algorithm terminates. Note that this heuristic will always construct at least a

Figure 1. Example Data and Solution

(a) An example problem instance					
j	0	1	2	3	4
t_{0j}	—	2	5	1	4
t_{1j}	1	—	3	7	3
t_{2j}	4	1	—	6	8
t_{3j}	4	1	4	—	5
t_{4j}	2	1	4	2	—
s_j	2	1	1	1	1
q_j	0	0.5	0.5	0.5	0.5
u_j	0	19	15	20	21



feasible solution (if any exists); in the worst case, each station is visited by an individual vehicle.

4.2. LNS for the JITCVRP-ext

First, we remove infeasible arcs from a given problem instance: An arc (i, j) for which the route $\langle 0, i, j, 0 \rangle$ violates the capacity restriction of the vehicles or the MFETs can never be part of a feasible solution.

Figure 2 shows a pseudocode overview of our solution method: First, we generate an initial solution, which is feasible with respect to the vehicle capacity and MFET by using the construction heuristic described above. To improve the generated solution, we use an LNS enhanced by an LS component. The purpose of the LNS component, which iteratively removes and re-adds parts of a solution using so-called destroy and repair operators, is to diversify the search (see Section 4.2.2 for a detailed description), while the LS serves intensification purposes (see Section 4.2.3). Contrary to many LNS implementations, our LNS is not restricted to feasible solutions; that is, solutions violating capacity constraints and MFET constraints are allowed during both the LNS and LS steps. The generalized cost function including penalties and the efficient evaluation of constraint violations are described in Section 4.2.1.

Figure 2. Overview of Our LNS Method

```

 $S \leftarrow \text{generateInitialSolution}()$ 
repeat
   $\delta \leftarrow \text{drawNumberOfStationsToRemove}()$ 
   $S' \leftarrow \text{Regret-Repair}(\text{Random-Remove}(S, \delta))$  {apply
    Random-Remove operator and repair solution with
    Regret-Repair}
   $S' \leftarrow \text{localSearch}(S')$ 
  if  $\text{ThresholdAccept}(f_{\text{gen}}(S'), f_{\text{gen}}(S))$  then
     $S \leftarrow S'$ 
  end if
until termination criteria satisfied

```

Solution acceptance is based on the threshold accepting scheme (Dueck and Scheuer 1990); that is, a solution S' is accepted if it either improves the incumbent solution S or does not deteriorate the incumbent solution by more than the current threshold T . We compare the objective function values of two solutions (and compute the difference between their objective function values) that were computed with different penalty factors for capacity and MFET violations by reevaluating the solutions with penalty factors that correspond to the average of the penalty factors that were used in the original evaluation (see Goeke and Schneider 2015). After each iteration, the threshold T is reduced to $(1 - r) \cdot T$, where $r = 0.01$ is the sink rate. The initial threshold T_0 is chosen such that T is equal to 1 after $\eta_{\text{noImpr}}^{\text{LNS}}/10$ iterations, that is, after a tenth of the number of iterations without improvement, after which the algorithm stops. In this way, we guarantee that in the first iterations of the heuristic, a broad range of solutions is accepted, while later on only improving solutions are accepted.

Note that most parameters of the algorithm are simply set to values similar to those found in the literature for related problems. No fine tuning of these parameters was conducted. The remaining parameters that are specific to the JITCVRP-ext (e.g., the penalty factor for violations of the MFET) were determined through preliminary experiments.

4.2.1. Generalized Cost Function. To increase the flexibility of the search, infeasible solutions with respect to vehicle capacity and MFET are allowed. We evaluate the objective value of a solution S by means of the following generalized cost function $f_{\text{gen}}(S)$:

$$f_{\text{gen}}(S) = \gamma \cdot m(S) + f(S) + \alpha \cdot P_{\text{cap}}(S) + \beta \cdot P_{\text{mfet}}(S),$$

where $m(S)$ denotes the number of routes (vehicles) in solution S , $f(S)$ denotes the total travel time (including service times), $P_{\text{cap}}(S)$ denotes the cumulative capacity

violation over all vehicle routes, and $P_{\text{mfet}}(S)$ denotes the violation of the MFET expressed as the cumulative lateness of the service at the stations. The constant factor γ assigns a weight to the number of vehicles. Because minimizing the number of employed tow trains is the primary goal of the optimization, we set $\gamma := \sum_{i \in V} \sum_{j \in V} t_{ij} + \sum_{i \in V} s_i$ to establish a lexicographic ordering of the two objectives. The penalty factors α and β are used for weighting the violations and are dynamically updated based on the course of the search. Both penalty factors are initially set to $0.5 \cdot \gamma$ and are restricted to the interval $[0.01 \cdot \gamma, 1.5 \cdot \gamma]$. After five consecutive iterations without a change in the feasibility status, the respective penalty factor is multiplied (divided) by 1.25 in case the solution is infeasible (feasible) with regard to the respective resource. From then on, the update is repeated after every iteration until the feasibility status changes.

When executing LNS and LS, a lot of different neighborhood solutions are evaluated. To enable fast processing, it is therefore crucial that the constraint violations of the neighboring solutions are quickly calculated. The capacity violations of a route generated by a station insertion during the LNS or by a search move in the LS can be efficiently calculated by saving forward and backward capacity slacks for all vertices. For classic neighborhood operators, including relocate, exchange, and 2-opt* as used in our LS, capacity violations for interroute moves can be determined in $\mathcal{O}(1)$ time. Contrary to the classic CVRP, the slacks in the JITCVRP-ext do not sum up absolute demands but the demand rates q_i . In addition, route duration slacks must be stored because the capacity on a route is determined by multiplying the summed up demand rates with the route duration.

Recall that we define a route π_k as a sequence of stations $\langle \pi_k(0), \pi_k(1), \dots, \pi_k(|R_k| + 1) \rangle$, with $\pi_k(0)$ and $\pi_k(|R_k| + 1)$ representing the depot 0. A solution S to a JITCVRP-ext instance is a set containing $m(S)$ routes $\{\pi_k: k = 1, \dots, m(S)\}$. Let $Vert(\pi_k)$ be the set of vertices that are part of route π_k , and let $Vert(S)$ be the set of vertices visited in a solution S .

We define the forward capacity slack and route duration slack of the vertices in a route π_k as follows:

$$\begin{aligned} \text{cap}_{\pi_k(0)}^{\rightarrow} &= 0, \\ \text{cap}_{\pi_k(i)}^{\rightarrow} &= \text{cap}_{\pi_k(i-1)}^{\rightarrow} + q_{\pi_k(i)}, \quad i = 1, \dots, |R_k| + 1; \\ \text{DUR}_{\pi_k(0)}^{\rightarrow} &= s_0, \\ \text{DUR}_{\pi_k(i)}^{\rightarrow} &= \text{DUR}_{\pi_k(i-1)}^{\rightarrow} + t_{\pi_k(i-1), \pi_k(i)} + s_{\pi_k(i)}, \quad i = 1, \dots, |R_k|, \\ \text{DUR}_{\pi_k(|R_k|+1)}^{\rightarrow} &= \text{DUR}_{\pi_k(|R_k|)}^{\rightarrow} + t_{\pi_k(|R_k|), 0}. \end{aligned}$$

The backward penalty slacks are computed analogously

$$\begin{aligned} \text{cap}_{\pi_k(|R_k|+1)}^{\leftarrow} &= 0, \\ \text{cap}_{\pi_k(i)}^{\leftarrow} &= \text{cap}_{\pi_k(i+1)}^{\leftarrow} + q_{\pi_k(i)}, \quad i = 0, \dots, |R_k|. \end{aligned}$$

$$DUR_{\pi_k(|R_k|+1)}^{\leftarrow} = 0,$$

$$DUR_{\pi_k(i)}^{\leftarrow} = DUR_{\pi_k(i+1)}^{\leftarrow} + t_{\pi_k(i), \pi_k(i+1)} + s_{\pi_k(i)}, \quad i = 0, \dots, |R_k|.$$

Now, let $\text{cap}_v^{\rightarrow}$, $\text{cap}_v^{\leftarrow}$, DUR_v^{\rightarrow} , and DUR_v^{\leftarrow} be known for all vertices $v \in Vert(S)$ of a solution S . Then, the capacity violation for LNS insertions and interroute LS moves can be calculated in constant time using the following two rules:

- If a route $\pi_{\text{new}} = \langle 0, \dots, x, y, \dots, 0 \rangle$ is constructed from two partial paths $\langle 0, \dots, x \rangle$ and $\langle y, \dots, 0 \rangle$, the capacity violation of the route can be determined as follows:

$$\begin{aligned} P_{\text{cap}}(\pi_{\text{new}}) &= \max\{(\text{cap}_x^{\rightarrow} + \text{cap}_y^{\leftarrow}) \\ &\quad \cdot (DUR_x^{\rightarrow} + DUR_y^{\leftarrow}) - Q, 0\}. \end{aligned}$$

- If a route $\pi_{\text{new}} = \langle 0, \dots, x, v, y, \dots, 0 \rangle$ is constructed by inserting a node v in between two partial paths $\langle 0, \dots, x \rangle$ and $\langle y, \dots, 0 \rangle$, the capacity violation of the new route can be calculated as follows:

$$\begin{aligned} P_{\text{cap}}(\pi_{\text{new}}) &= \max\{(\text{cap}_x^{\rightarrow} + \text{cap}_y^{\leftarrow} + q_v) \\ &\quad \cdot (DUR_x^{\rightarrow} + DUR_y^{\leftarrow} + s_v + t_{xv} + t_{vy}) - Q, 0\}. \end{aligned}$$

The next step is to find a way of efficiently calculating the cumulative violation of the MFETs in a route. Note that deciding the feasibility (yes/no) with regard to the MFETs in constant time is straightforward: we simply have to compare the total route duration (calculated using the route duration slacks described above) with the minimal value of the MFET of a station in the route. The difference between these two values could also be used as a penalty for the considered station insertion or LS move; however, the resulting value expresses the violation with regard to only the most critical station, and the number of violations throughout the route and their extents are not considered. Consequently, we consider the sum of the differences between the total route duration and the MFET for all stations. The complexity for calculating this penalty is $\mathcal{O}(\log n)$ if the following additional pieces of information are stored for each solution:

- For each station $\pi_k(i)$ on each route π_k , let $FREQ_{ki}^{\rightarrow}$ be a sorted list in ascending order of the MFETs on route k up to station i , that is, a sorted list of the elements in $\{u_{\pi_k(j)}: j = 1, \dots, i\}$. Let $FREQ_{ki}^{\rightarrow}(j)$ refer to the j th element in this list.

- For each station $\pi_k(i)$ on each route π_k , let $FREQ_{ki}^{\leftarrow}$ be a sorted list in ascending order of all MFETs on route k for all stations $i, \dots, |R_k|$, that is, a sorted list of the elements in $\{u_{\pi_k(j)}: j = i, \dots, |R_k|\}$. Let $FREQ_{ki}^{\leftarrow}(j)$ refer to the j th element in this list.

- Let $VIO_{ki}^{\rightarrow}(j)$ be the extent of the cumulative violation of the MFETs up to station i on route k if the route duration is equal to the j th element of $FREQ_{ki}^{\rightarrow}$; that is, $VIO_{ki}^{\rightarrow}(j) = \sum_{l=1}^{j-1} (FREQ_{ki}^{\rightarrow}(j) - FREQ_{ki}^{\rightarrow}(l))$, for all $j = 1, \dots, i$.

- Analogously, let $VIO_{ki}^{\leftarrow}(j)$ be the extent of the cumulative violation of the MFETs following station i on route k if the route duration is equal to the j th element of $FREQ_{ki}^{\leftarrow}$; that is, $VIO_{ki}^{\leftarrow}(j) = \sum_{l=1}^{j-1} (FREQ_{ki}^{\leftarrow}(j) - FREQ_{ki}^{\leftarrow}(l))$, for all $j = 1, \dots, |R_k| - i + 1$.

Given the described information, the change in the cumulative violation of the MFET can be determined using the following two rules:

- In the case that a route $\pi_{\text{new}} = \langle \pi_k(0), \dots, \pi_k(\underline{i}), \pi_{k'}(\bar{i}), \dots, \pi_{k'}(|R_{k'}| + 1) \rangle$ is constructed from two partial paths $\langle \pi_k(0), \dots, \pi_k(\underline{i}) \rangle$ and $\langle \pi_{k'}(\bar{i}), \dots, \pi_{k'}(|R_{k'}| + 1) \rangle$, we proceed as follows:

- Determine the duration of the new route:

$$dur_{\text{new}} = DUR_{\pi_k(\underline{i})}^{\rightarrow} + DUR_{\pi_{k'}(\bar{i})}^{\leftarrow}.$$

2. Determine the maximum index j_{\max}^{\rightarrow} of an entry for the MFET in $FREQ_{ki}^{\rightarrow}$ that is smaller than dur_{new} , that is, $j_{\max}^{\rightarrow} = \arg \max_{j=1, \dots, \underline{i}} \{FREQ_{ki}^{\rightarrow}(j) < dur_{\text{new}}\}$. This can be achieved in $\mathcal{O}(\log n)$ time using binary search on $FREQ^{\rightarrow}$. Analogously, let us denote $j_{\max}^{\leftarrow} = \arg \max_{j=1, \dots, |R_{k'}| - \bar{i} + 1} \{FREQ_{k' \bar{i}}^{\leftarrow}(j) < dur_{\text{new}}\}$.

- If there exists no station whose MFET is violated by dur_{new} , then $P_{\text{mfet}}(\pi_{\text{new}})$ is 0; otherwise

$$\begin{aligned} P_{\text{mfet}}(\pi_{\text{new}}) = & VIO_{ki}^{\rightarrow}(j_{\max}^{\rightarrow}) + (dur_{\text{new}} - FREQ_{ki}^{\rightarrow}(j_{\max}^{\rightarrow})) \cdot j_{\max}^{\rightarrow} \\ & + VIO_{k' \bar{i}}^{\leftarrow}(j_{\max}^{\leftarrow}) + (dur_{\text{new}} - FREQ_{k' \bar{i}}^{\leftarrow}(j_{\max}^{\leftarrow})) \cdot j_{\max}^{\leftarrow}. \end{aligned}$$

- If a route $\pi_{\text{new}} = \langle 0, \dots, \pi_k(\underline{i}), v, \pi_{k'}(\bar{i}), \dots, 0 \rangle$ is constructed by inserting a node v in between two partial paths $\langle 0, \dots, \pi_k(\underline{i}) \rangle$ and $\langle \pi_{k'}(\bar{i}), \dots, 0 \rangle$, the MFET violation of the route can be determined as follows:

- Determine the duration of the new route:

$$dur_{\text{new}} = DUR_{\pi_k(\underline{i})}^{\rightarrow} + DUR_{\pi_{k'}(\bar{i})}^{\leftarrow} + s_v + t_{\pi_k(\underline{i}), v} + t_{v, \pi_{k'}(\bar{i})}.$$

- Determine the maximum indices

$$j^{\rightarrow} = \arg \max_{j=1, \dots, \underline{i}} \{FREQ_{ki}^{\rightarrow}(j) < dur_{\text{new}}\} \quad \text{and}$$

$$j^{\leftarrow} = \arg \max_{j=1, \dots, |R_{k'}| - \bar{i} + 1} \{FREQ_{k' \bar{i}}^{\leftarrow}(j) < dur_{\text{new}}\}.$$

- If there exists no station whose MFET is violated by dur_{new} , then $P_{\text{mfet}}(\pi_{\text{new}})$ is 0; otherwise

$$\begin{aligned} P_{\text{mfet}}(\pi_{\text{new}}) = & VIO_{ki}^{\rightarrow}(j^{\rightarrow}) + (dur_{\text{new}} - FREQ_{ki}^{\rightarrow}(j^{\rightarrow})) \cdot j^{\rightarrow} \\ & + VIO_{k' \bar{i}}^{\leftarrow}(j^{\leftarrow}) + (dur_{\text{new}} - FREQ_{k' \bar{i}}^{\leftarrow}(j^{\leftarrow})) \cdot j^{\leftarrow} \\ & + \max\{0; dur_{\text{new}} - u_v\}. \end{aligned}$$

Figure 3. Penalty Slacks in the Example

	0	3	2	1	0
DUR $^{\rightarrow}$	2	4	9	11	12
DUR $^{\leftarrow}$	12	9	4	2	0
CAP $^{\rightarrow}$	0	0.5	1	1.5	1.5
CAP $^{\leftarrow}$	1.5	1.5	1	0.5	0
FREQ $^{\rightarrow}$		$\langle 20 \rangle$	$\langle 15, 20 \rangle$	$\langle 15, 19, 20 \rangle$	
FREQ $^{\leftarrow}$		$\langle 15, 19, 20 \rangle$	$\langle 15, 19 \rangle$	$\langle 19 \rangle$	
VIO $^{\rightarrow}$		$\langle 0 \rangle$	$\langle 0, 5 \rangle$	$\langle 0, 4, 6 \rangle$	
VIO $^{\leftarrow}$		$\langle 0, 4, 6 \rangle$	$\langle 0, 4 \rangle$	$\langle 0 \rangle$	

Consider the example from Section 3.2. Moreover, consider a route $\langle 0, 3, 2, 1, 0 \rangle$. The forward and backward penalty slacks for this route are shown in Figure 3. Now, assume that station 4 is to be inserted between stations 2 and 1; that is, $\underline{i} = 2$ and $\bar{i} = 3$. First, we calculate the penalty value for the capacity violation as $P_{\text{cap}} = \max\{(1 + 0.5 + 0.5) \cdot (9 + 2 + 1 + 8 + 1) - 20, 0\} = 22$. Second, we get the penalty value for the MFET violation by following the three steps:

- Determine the duration of the new route: $dur_{\text{new}} = DUR_{\pi_k(\underline{i})}^{\rightarrow} + DUR_{\pi_{k'}(\bar{i})}^{\leftarrow} = 9 + 2 + 1 + 8 + 1 = 21$.
- Determine the maximum indices $j^{\rightarrow} = 2$ and $j^{\leftarrow} = 1$.
- There is at least one station whose MFET is violated; hence,

$$\begin{aligned} P_{\text{mfet}} = & 5 + (21 - 20) \cdot 2 \\ & + 0 + (21 - 19) \cdot 1 \\ & + \max\{0; 21 - 21\} \\ = & 9. \end{aligned}$$

4.2.2. Large Neighborhood Search. LNS was originally introduced by Shaw (1998) and extended to adaptive LNS (ALNS) by Ropke and Pisinger (2006). The idea is to iteratively destroy and repair larger parts of an initial solution to gradually improve the solution. ALNS has successfully been applied to many hard to solve combinatorial optimization problems (Pisinger and Ropke 2010).

Our LNS uses Random-Remove as a destroy operator because of the very good diversification capabilities of this operator. We determine the number of stations to remove δ in an LNS iteration by randomly drawing δ from the interval $[0.05n, 0.6n]$. As a repair operator, Regret-2 is used. The difference in cost increase for the insertion of all vertices currently not included in the solution at their best and at their second best position is calculated, and the vertices are stored in a list in descending order of this regret value. To diversify the search, instead of selecting the first vertex in the list for insertion, one of the first 15 vertices is randomly drawn with uniform probability.

The LNS runs for a maximum of $\eta_{\text{tot}}^{\text{LNS}} = 10,000$ iterations but is stopped earlier if no improving solution is found for $\eta_{\text{holmp}}^{\text{LNS}} = 500$ consecutive iterations.

4.2.3. Local Search. The solution returned by the LNS step is improved by an LS that is run for a maximum of $\eta_{\text{tot}}^{\text{LS}} = 1,000$ iterations but stopped earlier if no improving solution can be found in an iteration. LS uses a composite neighborhood of relocate (Savelsbergh 1992), exchange (Savelsbergh 1992), and 2-opt* (Potvin et al. 1996). The first two operators are used for intra- and interroute moves; 2-opt* is defined only for interroute moves.

5. Variants of the JITCVRP-ext

In this section, we discuss how to extend our LNS to handle split deliveries (Section 5.1) or a special layout of the shop floor in which all stations are located on a line (Section 5.2).

5.1. Split Deliveries

One of our assumptions when modeling and solving the JITCVRP-ext is that each station is visited on exactly one route, and the entire demand of the station is satisfied on that visit. While this is certainly often the case because of practical expediency, there is no fundamental reason against serving the same station on more than one route. If the demand of a station exceeds the maximum vehicle capacity, it is even necessary to do so. The resulting problem belongs to the class of split-delivery VRPs, which are notoriously hard to solve, even without the presence of MFET (see, e.g., Archetti and Speranza 2008, 2012; Irnich, Schneider, and Vigo 2014).

To extend the LNS to be able to handle split deliveries, we follow the idea of Chen et al. (2017): The authors describe a technique to use existing VRP algorithms to solve the split-delivery VRP with very good results. The basic idea of their approach is to split the demand of each customer a priori; that is, each customer is replaced with a set of dummy customers, who are physically in the same location but are each associated only with a fraction of the demand of the original customer. The total demand of the dummy customers is equal to the original demand. The resulting problem instance can then be solved with the original CVRP algorithm.

For solving the JITCVRP-ext with split deliveries, we adapt this idea in the following way. First, we determine the station with the highest demand rate $q^* = \max_{i \in V \setminus \{0\}} \{q_i\}$. Then, for each station $i \in V \setminus \{0\}$, we break down the demand rate q_i into a_{20} pieces of $0.2 \cdot q^*$, a_{10} pieces of $0.1 \cdot q^*$, a_5 pieces of $0.05 \cdot q^*$, and one piece of the remaining demand as follows:

$$a_{20} = \arg \max_{a \in \mathbb{N}^0} \{0.2 \cdot q^* \cdot a \leq q_i\},$$

$$a_{10} = \arg \max_{a \in \mathbb{N}^0} \{0.1 \cdot q^* \cdot a \leq q_i - 0.2 \cdot q^* \cdot a_{20}\},$$

$$a_5 = \arg \max_{a \in \mathbb{N}^0} \{0.05 \cdot q^* \cdot a \leq q_i - 0.2 \cdot q^* \cdot a_{20} - 0.1 \cdot q^* \cdot a_{10}\}.$$

The original station is replaced by dummy stations with the calculated fractional demand rates, and the resulting instance stays a standard JITCVRP-ext instance, which can be solved using our LNS.

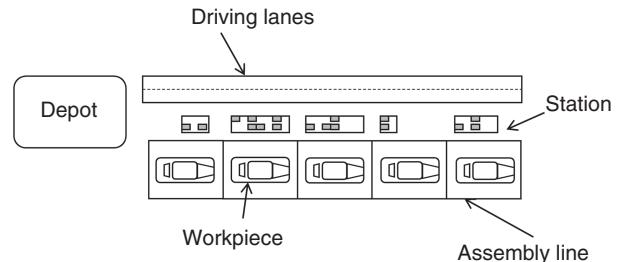
5.2. Stations Lying on a Line

Our JITCVRP-ext model and our LNS do not make any assumptions about the layout within the manufacturing plant; that is, stations can be sited arbitrarily on the shop floor. This allows us to flexibly model a broad range of applications. However, in many assembly plants, the floor layout follows a strict pattern. Specifically, stations must be located along the assembly line, which often runs in a fairly straight line with few bends. Assume that all stations lie on a straight line, with the depot being located on one side of the line. It stands to reason that, given such a specific layout, the JITCVRP-ext becomes somewhat easier to solve. Emde and Boysen (2012b) have investigated a similar layout, albeit under the additional assumption that stations on a route must always be contiguous (e.g., if stations 1 and 3 are on the same route, then so must be station 2).

A layout with all stations lying on a straight line is depicted schematically in Figure 4. Clearly, in this case, the choice of sensible routes is limited compared to the general case, in which stations may be anywhere on the shop floor: it can never make sense for a tow train to zigzag between stations; instead, the vehicle should go straight to the station on the route that is the farthest from the depot, visit the other assigned stations as it passes them along the way, then go straight back to the depot. However, this does not mean that the JITCVRP-ext becomes trivial under the straight-line assumption. The decision of which stations to assign to which route remains, which is a strongly NP-hard problem. Although the routing aspect becomes easier, the stations still have different service times s_i and demand rates q_i , which already makes the assignment of stations to vehicles quite challenging. Note also that the MFET constraints continue to be relevant and difficult constraints, regardless of whether a straight line or general shop-floor layout is assumed.

Nonetheless, the problem certainly becomes easier because some edges can be removed from the input

Figure 4. A Straight Assembly Line



digraph G . As described above, zigzagging makes no sense on a straight line, and therefore all arcs $(i, j) \in A$ where $t_{0i} > t_{0j}$ can be removed from G . In addition, this property can be used to strongly reduce the number of potential moves that need to be evaluated in the local search.

6. Computational Study

This section presents the numerical studies to assess the performance of our LNS. In Section 6.1, we describe the generation of new benchmark instances and the properties of the benchmark sets available from the literature. In Section 6.2, we investigate the effects of the different components of our LNS, and Section 6.3 compares the results of our LNS to those of the nearest neighbor heuristic and a commercial solver. Section 6.4 studies the practical ramifications of assuming constant demand rates. Finally, we investigate the computational efficacy of our LNS heuristic for the two variants of the JITCVRP-ext with split deliveries (Section 6.5) and a layout with stations located on a straight line (Section 6.6).

6.1. Benchmark Instances and Computational Environment

For our computational tests, we use two different instance sets: (i) a purely random, newly generated set is used to study the performance of our LNS and (ii) a set from the literature that is based on industry data from a major German car manufacturer. We describe both data sets in more detail below. Note that in the following, we measure all time related parameters in time units, where one TU corresponds to one work cycle in the automotive industry, which is the inverse of the production rate. Similarly, the distance related parameters are measured such that one duration unit corresponds to the distance a tow train travels during one work cycle. In a typical assembly plant for compact cars, a work cycle is equivalent to about 60 to 90 seconds (Emde, Fliedner, and Boysen 2012), and according to Faccio et al. (2013), the average speed of a tow train is about 0.8 m/s, considering blockages, bends, and other obstacles. Consequently, a tow train may cover a distance of about 60 m during a work cycle.

First, we create three random data sets, each containing 10 instances, one featuring small instances ($n = 12$ stations), one medium instances ($n = 50$ stations), and one large instances ($n = 300$ stations). We assign to each station i (and the depot $i = 0$) a two-dimensional coordinate (x_i, y_i) , with $x_i = \text{rnd}(0.1; 10)$ and $y_i = \text{rnd}(0.1; 10)$, where rnd denotes a random real number from the interval in the argument. The driving time between two distinct stations i and j is then calculated as the Euclidean distance between these stations, that is, $t_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. For each station

$i \in V \setminus \{0\}$, we set the service time to $s_i = \text{rnd}(0.1; 4)$. The service time at the depot is $s_0 = \text{rnd}(1; 10)$. Moreover, we set the demand rate to $q_i = \text{rnd}(0.05; 0.1)$ bins per TU. The MFETs are set to $u_i = (1 + \text{rnd}(0; 15)) \cdot (s_0 + s_i + t_{0i} + t_{i0})$, and the vehicle capacity is $Q = 20 \cdot \max_{i \in V \setminus \{0\}} \{(s_0 + s_i + t_{0i} + t_{i0}) \cdot q_i\}$.

Note that while these instances are randomly generated primarily for the purpose of testing the computational performance of our algorithm and model, the parameter ranges are in the ballpark of what we have observed at automotive assembly plants in Germany.

As a second benchmark set, we use the one proposed by Emde and Gendreau (2017). The authors use the data to optimize schedules for tow trains in the context of final assembly at a major German automotive company. Because the problem tackled in Emde and Gendreau (2017) consists of determining the short-term timetables given fixed routes, the production sequence is already established, and hence, the demand in terms of standard-size bins at the stations is known with certainty. To convert the discrete demands derived from a specific production sequence to a demand rate q_i , we take the arithmetic mean. For example, if the data say that at some station i a total of 36 bins are in demand during the planning horizon of 144 work cycles (one shift), we set $q_i = 36/144 = 0.25$, that is, on average, one bin per four cycles is in demand. Similarly, we determine the MFET u_i of a station i by dividing the actual rack space of that station (as reported by Emde and Gendreau 2017) by q_i . For example, if the rack at station i can hold eight bins and the demand rate as calculated above is $q_i = 0.25$, we get $u_i = 8/0.25 = 32$; that is, the tow train must visit station i at least once per 32 cycles. All other parameters (service times, driving times, vehicle capacity) can be copied without further conversion from the original data.

The original data set by Emde and Gendreau (2017) contains 10 small ($n = 10$ stations, planning horizon of 10 work cycles) and 10 large ($n = 20$ stations, 144 work cycles) instances, yielding a total of 50 instances used in our computational study. In Sections 6.2–6.5, we denote the randomly generated instances as “R k ,” where k is the instance number: R1–R10 are the small instances, R11–R20 are the medium-size instances, and R21–R30 are the large ones. Analogously, EG1–EG10 (EG11–EG20) denote the small (large) instances from Emde and Gendreau (2017). All instances are available from the authors on request.

We implemented all algorithms in C# 6.0 and conducted the numerical studies on an x64 PC equipped with an Intel Core i7-6700K 4 GHz CPU and 64 GB RAM. We restricted our heuristics to one CPU core, whereas the default solver (CPLEX 12.6.3) was left at its default setting, using all available cores.

6.2. Analysis of the Components of the LNS Algorithm

Our LNS algorithm is composed of three individual components that contribute to its performance: the LNS component, the local search, and the calculation of MFET constraint violations. To analyze which parts are the most important ones, we tested four different versions of our LNS: the three versions beside the full LNS are obtained by “deactivating” each of the three main components described above. Note that we only deactivate one single component per version of the algorithm, never multiple components at once. More specifically, we make the following modifications:

- *Version noPen*: As discussed in Section 4.2.1, it is easy to check the feasibility (yes/no) in constant time. In this version of our LNS, we replace our logarithmic time scheme for calculating the precise extent of MFET constraint violation with a simple constant time feasibility check, which merely tests whether a given solution is feasible or not, returning either 1 (solution infeasible) or 0 (solution feasible). Note that the generalized cost function and the dynamic penalty parameter update are otherwise unchanged.
- *Version noLS*: This version does not contain the local search component.
- *Version noLNS*: This version does not contain the destroy and repair phase.

We conducted 10 runs of each of the four algorithm versions on the medium-sized ($n = 50$) and large ($n = 300$) instances introduced above. Table 2 shows the results of the comparison. For each of the three variants with deactivated components, we report for each instance the number of vehicles Δm and the total route duration Δf of the best of the 10 runs as the gap to the best of the 10 runs of the full LNS. More precisely, Δm is reported as absolute gap $\Delta m = m_V - m_F$, with m_V the number of vehicles in the best solution obtained with the tested version and m_F the number of vehicles in the best solution obtained with the full LNS. Analogously, Δf is given as percentage gap $\Delta f = (f_V - f_F)/f_F$.

The tests reveal that all three components contribute significantly to the ultimate solution quality. Using our logarithmic feasibility test, the number of vehicles employed and the total route duration drop by 0.3 and 0.43% on average, respectively. The gap is even greater for the other components. The local search especially is apparently quite powerful. Its deactivation often leads to solutions requiring multiple additional vehicles.

6.3. Comparison of Solution Methods

To investigate the performance of our full LNS heuristic, we conducted 10 runs on the 30 newly generated random test instances. Table 3 reports the results on the 10 small instances compared with (i) CPLEX 12.6.3 solving the MIP model from Section 3 and (ii) the nearest neighbor construction heuristic from Section 4.1.

Table 2. Effect of the Main Components of LNS on the Solution Quality for Medium-Sized and Large Instances

Instance	noPen		noLS		noLNS	
	Δm	Δf (%)	Δm	Δf (%)	Δm	Δf (%)
R11	0	0.00	1	17.78	1	4.32
R12	0	0.05	2	25.07	1	4.79
R13	0	0.93	3	23.47	0	2.77
R14	0	0.70	1	18.06	1	19.43
R15	0	0.12	1	10.33	0	0.12
R16	0	-0.13	1	22.80	1	5.74
R17	0	0.04	3	32.09	2	4.80
R18	1	0.49	1	11.86	1	3.33
R19	0	0.00	2	19.36	1	5.34
R20	0	1.44	1	19.94	0	1.88
R21	1	-0.06	8	16.72	2	1.96
R22	0	-0.04	7	19.36	1	0.63
R23	1	0.89	7	18.77	3	3.02
R24	1	1.48	9	20.80	1	1.54
R25	-1	-0.84	6	14.23	2	2.39
R26	1	-0.20	7	17.05	1	0.39
R27	0	-0.24	9	18.89	2	1.17
R28	1	1.21	7	16.07	2	1.97
R29	0	0.95	7	18.93	1	2.28
R30	1	1.91	7	19.52	2	2.16
Avg.	0.30	0.43	4.50	19.05	1.25	3.50

For each solution method, we report the number of vehicles m and total route duration f . For LNS, the average and the best solution quality of the 10 runs are reported as m_a and f_a and m_b and f_b , respectively. For CPLEX and LNS, the CPU times in seconds are reported in the table; in the case of LNS, the times listed are the average CPU times of the 10 runs. NN could solve all instances with negligible effort, and therefore the CPU time is not listed.

The results show that LNS is capable of solving small instances to optimality in a short time. It finds the optimal solution to all small instances in under one second of CPU time per run, whereas CPLEX struggles with many of these problems, exceeding one hour of CPU time in one instance. Note that we also experimented with slightly larger instance sizes but found that $n = 12$ stations is just about CPLEX’s limit. With the medium-size instances ($n = 50$), no useful bounds—let alone optimal solutions—can be found. The simple NN heuristic is very quick but delivers decidedly suboptimal results. In four cases, it could not even find a solution with the minimum number of routes.

On the large instances (Table 4), LNS can substantially improve on NN, saving 7.4 vehicles and 234.2 units of total route duration on average. The same also holds for the medium-size instances, where LNS can save about 1.6 vehicles in under seven seconds per run on average. Even for the large instances, the average CPU time of LNS is approximately seven minutes

Table 3. Comparison of the Results Obtained by CPLEX, by the Nearest Neighbor Heuristic, and Our LNS on the Small ($n = 12$) Instances

Instance	CPLEX			NN		LNS				
	<i>m</i>	<i>f</i>	CPU sec.	<i>m</i>	<i>f</i>	<i>m_b</i>	<i>m_a</i>	<i>f_b</i>	<i>f_a</i>	CPU sec.
R1	3	79.0	639.1	3	92.6	3	3	79.0	79.3	0.7
R2	2	72.6	3,231.6	2	76.3	2	2	72.6	72.6	0.6
R3	2	67.5	390.3	3	83.3	2	2	67.5	67.5	0.7
R4	2	79.6	232.9	3	107.8	2	2	79.6	79.6	0.7
R5	2	89.4	4,943.2	2	96.5	2	2	89.4	89.4	0.7
R6	3	75.5	82.6	3	81.3	3	3	75.5	75.5	0.8
R7	3	68.4	19.5	3	73.5	3	3	68.4	68.4	0.8
R8	2	81.2	1,702.4	3	99.4	2	2	81.2	81.2	0.7
R9	2	93.7	3,319.8	2	105.0	2	2	93.7	93.7	0.6
R10	2	75.2	968.6	3	95.1	2	2	75.2	75.2	0.6
Avg.	2.3	78.2	1,553.0	2.7	91.1	2.3	2.3	78.2	78.3	0.7

Note. The optimal objective value per instance is in bold.

per run, which is clearly acceptable for a medium-term problem like the JITCVRP-ext. Note also that the results are fairly stable in between runs: The average number of vehicles in 10 runs is only 0.1, for medium-size instances, and 1, for large instances, more than the best number of vehicles, and the relative deterioration of route duration is marginal. Moreover, the average results are still substantially better than with NN, both in terms of the number of vehicles and route duration.

Apart from these random instances, we also tested the algorithms on 20 instances from the literature. The corresponding results are shown in Tables 5 and 6. As expected, CPLEX is not able to solve the larger instances with 20 stations, but can solve all 10-station instances to optimality, partly requiring several minutes of run time. For the small instances, LNS finds the optimal solution in all cases. NN, on the other hand, finds the optimal solution in only three cases, which are

Table 4. Comparison of the Results Obtained by the Nearest Neighbor Heuristic and LNS on the Medium-Size ($n = 50$) and Large ($n = 300$) Instances

Instance	NN			LNS				
	<i>m</i>	<i>f</i>	<i>m_b</i>	<i>m_a</i>	<i>f_b</i>	<i>f_a</i>	CPU sec.	
R11	7	325.8	6	6.0	275.4	279.6	6.7	
R12	8	286.1	6	6.0	228.3	231.2	6.3	
R13	10	285.1	7	7.0	229.9	233.4	7.9	
R14	8	241.9	7	7.0	202.5	205.0	6.3	
R15	9	250.3	8	8.0	225.5	226.0	7.1	
R16	7	286.4	6	6.0	231.3	233.8	6.8	
R17	10	282.1	7	7.3	211.5	216.3	7.5	
R18	8	285.6	7	7.8	253.5	257.4	6.4	
R19	9	305.4	7	7.0	252.5	254.8	7.3	
R20	8	273.9	7	7.0	225.5	228.8	6.3	
Avg.	8.4	282.3	6.8	6.9	233.6	236.6	6.9	
R21	42	1,379.4	34	35.6	1,175.8	1,197.9	485.3	
R22	36	1,572.7	29	29.8	1,310.6	1,322.7	470.5	
R23	39	1,434.0	32	33.0	1,203.4	1,219.6	455.5	
R24	43	1,509.5	34	34.6	1,245.5	1,262.5	442.6	
R25	33	1,447.5	27	27.7	1,224.7	1,239.0	458.8	
R26	39	1,507.2	32	33.3	1,284.5	1,292.9	440.4	
R27	45	1,465.2	36	37.0	1,221.4	1,229.1	451.7	
R28	40	1,273.7	33	34.4	1,091.9	1,115.0	483.0	
R29	41	1,529.4	34	34.4	1,279.2	1,293.9	441.6	
R30	36	1,563.4	29	30.2	1,303.1	1,323.2	427.3	
Avg.	39.4	1,468.2	32	33.0	1,234.0	1,249.6	455.7	

Note. The best objective value per instance is in bold.

Table 5. Comparison of the Results Obtained by CPLEX, by the Nearest Neighbor Heuristic, and LNS for the Small ($n = 10$) Instances from the Literature

Instance	CPLEX			NN			LNS			
	<i>m</i>	<i>f</i>	CPU sec.	<i>m</i>	<i>f</i>	<i>m_b</i>	<i>m_a</i>	<i>f_b</i>	<i>f_a</i>	CPU sec.
EG1	2	13.8	219.3	2	15.6	2	2	13.8	14.5	0.4
EG2	2	14.6	107.2	2	16.6	2	2	14.6	14.6	0.5
EG3	2	14.0	266.6	2	16.0	2	2	14.0	14.0	0.5
EG4	1	10.6	14.1	1	10.6	1	1	10.6	10.6	0.5
EG5	1	10.4	17.7	1	10.4	1	1	10.4	10.4	0.5
EG6	2	14.2	129.9	2	16.4	2	2	14.2	14.2	0.5
EG7	2	14.8	136.4	2	17.0	2	2	14.8	14.8	0.6
EG8	2	13.8	290.5	2	16.0	2	2	13.8	13.8	0.5
EG9	2	13.4	220.1	2	15.2	2	2	13.4	14.1	0.4
EG10	1	10.8	15.9	1	10.8	1	1	10.8	10.8	0.6
Avg.	1.7	13.0	141.8	1.7	14.5	1.7	1.7	13.1	13.2	0.5

Note. The optimal value per instance is in bold.

apparently trivial, judging by the run times of CPLEX on those instances. The larger instances with 20 stations turn out to be also trivial to solve; NN finds the same solutions as LNS for all instances. The vehicle capacity and MFETs seem to be so generous that it is always possible to supply all stations with only one vehicle.

6.4. Effect of Time-Varying Demand Rates

In this part of our computational study, we investigate whether our assumption of constant demand rates impacts the day-to-day performance of the part-feeding process, in which actual part demands are not constant or real valued. To this end, we reuse the instance set of Emde and Gendreau (2017). Originally, these instances use discrete demand data that can be derived from the production sequence and the bill of materials, detailing exactly how many bins are in demand at what station at what time. By averaging these demands, we have calculated the constant

demand rates, which constitute the input for the JITCVRP-ext (see Section 6.1).

Given the actual discrete demand at the stations and the routes returned by LNS on the basis of the average demand rates, we can calculate the exact amount delivered to each station as follows. We assume—in line with our JITCVRP-ext assumptions—that routes are toured cyclically without break from the beginning of the planning horizon and that bins are brought to the respective stations as just-in-time as possible. This means that if the tow train arrives at some station s at time t and then again, on its next tour, at some later time t' , it will deliver to station s exactly the number of bins in demand between t and t' . Scheduling deliveries on the given routes in this way guarantees that all demand will always be satisfied; however, it is possible that the vehicle or station capacities (MFETs) are exceeded. A feasible JITCVRP-ext solution ensures only that the capacities are not violated *on average*. However, given specific time-varying demands, there may still be problems in some periods.

Table 7 lists the total number of bins that exceed the vehicle and/or station capacities (MFETs) for each of the instances of Emde and Gendreau (2017) when routes are generated based on average demands. For our test data, the vehicles never have to be overloaded to ensure just-in-time supply to the assembly line. However, occasionally the shelf space of a few stations is slightly exceeded, indicating that adhering to MFETs on average is not enough to guarantee feasibility in the daily operation of the tow train system. Nonetheless, the violations are comparatively minor. The small instances consist of 10 stations over a planning horizon of 10 cycles, and yet the total excess of the station capacity in the worst instance is only three bins. It stands to reason that this could be rectified even without changing the routes just by employing a more sophisticated schedule than simple cyclical repetition (see Emde and

Table 6. Comparison of the Results Obtained by the Nearest Neighbor Heuristic and LNS for the Large ($n = 20$) Instances from the Literature

Instance	NN			LNS			
	<i>m</i>	<i>f</i>	<i>m_b</i>	<i>m_a</i>	<i>f_b</i>	<i>f_a</i>	CPU sec.
EG11	1	27.2	1	1	27.2	27.2	1.18
EG12	1	27.6	1	1	27.6	27.6	1.07
EG13	1	27.6	1	1	27.6	27.6	1.09
EG14	1	27.4	1	1	27.4	27.4	1.31
EG15	1	27.2	1	1	27.2	27.2	1.20
EG16	1	27.2	1	1	27.2	27.2	1.15
EG17	1	27.0	1	1	27.0	27.0	1.07
EG18	1	27.8	1	1	27.8	27.8	1.08
EG19	1	27.2	1	1	27.2	27.2	1.13
EG20	1	27.4	1	1	27.4	27.4	1.21
Avg.	1	27.4	1	1	27.4	27.4	1.15

Note. The best value per instance is in bold.

Table 7. Violations of Vehicle and Station Capacities (MFETs) Caused by Assuming Constant Demand Rates

Instance	Vehicle capacity	Station capacity
EG1	0	0
EG2	0	3
EG3	0	1
EG4	0	2
EG5	0	1
EG6	0	1
EG7	0	0
EG8	0	1
EG9	0	1
EG10	0	1
EG11	0	0
EG12	0	0
EG13	0	0
EG14	0	0
EG15	0	0
EG16	0	0
EG17	0	0
EG18	0	0
EG19	0	0
EG20	0	0

Gendreau 2017). Alternatively, it may be expedient to tighten the MFET constraint a little and generate new routes via LNS. Note that in the large instances with 20 stations and 144 cycles, there is not a single violation, probably because the capacities are more generous in those instances.

6.5. Effect of Split Deliveries

We solved the medium-sized instances with $n = 50$ stations allowing for split deliveries using the procedure described in Section 5.1. Table 8 compares the results to those obtained without split deliveries: m refers to the number of vehicles of the best solution found in 10 runs, and f to the route duration of this solution.

As expected, allowing split deliveries can slightly improve the solution quality because of the additional flexibility. While the number of vehicles is the same

Table 8. Results With and Without Split Deliveries for Medium-Sized Instances

Instance	Split		No split	
	m	f	m	f
R11	5	238.7	6	275.4
R12	6	217.9	6	228.3
R13	7	229.4	7	229.9
R14	7	201.5	7	202.5
R15	8	221.7	8	225.5
R16	6	208.0	6	231.3
R17	7	211.4	7	211.5
R18	7	242.8	7	253.5
R19	7	243.3	7	252.5
R20	7	224.6	7	225.5
Avg.	6.7	223.9	6.8	233.6

Table 9. Comparison of the Average Results Obtained for the Instance Sets Assuming That the Stations Are Located on a Straight Line and the Instances Not Restricting the Locations of Stations

n	LNS (line)			LNS		
	m	f	CPU sec.	m	f	CPU sec.
12	2.0	193.4	0.6	2.3	78.3	0.7
50	4.4	1,216.2	5.4	6.9	236.6	6.9
300	12.4	18,848.6	344.3	33.0	1,249.6	455.7

in all instances but one, there is a nonnegligible drop of about 4% in the average total duration of the tours. However, given that split deliveries make communicating and coordinating trips harder and carry the risk of blockages if multiple vehicles want to access the same station at the same time, the gap may not be large enough to justify allowing them in practice—at least not if station demands exceeding the vehicle capacity do not occur.

6.6. Effect of Stations Lying on a Straight Line

To investigate whether the simplified floor layout found in many assembly plants makes solving the JITCVRP-ext easier, we generate a new set of instances exactly as described in Section 6.1, except that all stations lie on a straight line in front of the depot. Specifically, station $i \in V \setminus \{0\}$ is located at point $(x_{i-1} + rnd(0.1; 10), 0)$, where $(0, 0)$ is the coordinate of the depot ($i = 0$).

Table 9 shows the average results over all small, medium, and large instances if all stations lie on a line and in the general case. The number of vehicles and route duration cannot be directly compared because the instances are different. However, we can compare the run times of LNS: pruning arcs from the graph and restricting local search moves in the case of a straight line layout noticeably speeds up the solution process. For the large instances, the average relative drop in CPU time is about 32%. This suggests that it is certainly worthwhile to give some thought to tuning the arc set and local search operators to specific floor layouts.

7. Conclusion

In this paper, we investigated the problem of planning routes for in-house milk-run deliveries of parts to stations in an assembly plant. To this end, we extended a problem from the literature, modeled it as a mixed-integer linear program, and proposed a novel large neighborhood search scheme to solve it. The main results are as follows.

- One of the main characteristics that distinguishes the JITCVRP-ext from previous routing problems is that adding stations to a route decreases the service frequency for *all* stations on the route. We proposed a

scheme to evaluate the violation of the maximum feasible elapsed time in logarithmic time in this paper.

• The LNS heuristic vastly outperforms the default solver we used, which proved to be incapable of solving instances with more than 12 stations in acceptable time. The LNS also produces a negligible optimality gap on small instances and significantly better results than a simple nearest neighborhood construction heuristic for large instances, all the while maintaining moderate CPU times.

• Assuming constant demand rates is a simplification because in daily operation, demands will of course be discrete. Using test data from the literature, we show that solving the JITCVRP-ext with constant demand rates can indeed sometimes lead to some violations of station capacities; however, these violations are usually small.

• Allowing split deliveries is able to only very slightly improve the solution quality. Given the possible negative consequences like harder communication and coordination and the risk of blockages, split deliveries seem only worthwhile if station demands exceed the vehicle capacity.

Future research could focus on our assumption that service times at the stations are independent of the number of bins to be (un)loaded. In (partially) automated systems, this is certainly adequate; however, in fully manual tow train delivery systems, a human operator may indeed need more time to swap extra bins. Moreover, it may be worthwhile to consider delivery patterns per vehicle (e.g., certain stations are not served on each trip but only periodically) such that a decision must also be made on when a vehicle should visit which station.

References

- Aghezzaf EH, Raa B, Van Landeghem H (2006) Modeling inventory routing problems in supply chains of high consumption products. *Eur. J. Oper. Res.* 169(3):1048–1063.
- Akiloglu A, Baydoğan M, Bolatlı Y, Canbaz D, Halıcı A, Sezgin Ö, Özdemirel N, Türkcan A (2006) Dizel enjektör üretimi yapan bir şirket için fabrika içi çekme esaslı tekrarlı dağıtım sistemi tasarımları [Pull-based milk-run distribution system design for a firm producing diesel injectors]. *Endüstri Mühendisliği Dergisi* 17(3):2–15.
- Archetti C, Speranza MG (2008) The split delivery vehicle routing problem: A survey. Golden B, Raghavan S, Wasil E, Sharda R, Voß S, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Oper. Res./Comput. Sci. Interfaces Series, Vol. 43 (Springer, Boston), 103–122.
- Archetti C, Speranza MG (2012) Vehicle routing problems with split deliveries. *Internat. Trans. Oper. Res.* 19(1–2):3–22.
- Baita F, Ukovich W, Pesenti R, Favaretto D (1998) Dynamic routing-and-inventory problem: A review. *Transportation Res. Part A* 32(8):585–598.
- Battini D, Boysen N, Emde S (2013) Just-in-time supermarkets for part supply in the automobile industry. *J. Management Control* 24(2):209–217.
- Beasley J (1984) Fixed routes. *J. Oper. Res. Soc.* 35(1):49–55.
- Boysen N, Fliedner M, Scholl A (2009) Level scheduling under storage constraints. *Internat. J. Production Res.* 47(10):2669–2684.
- Boysen N, Emde S, Hoeck M, Kauderer M (2015) Part logistics in the automotive industry: Decision problems, literature review and research agenda. *Eur. J. Oper. Res.* 242(1):107–120.
- Campbell A, Clarke L, Kleywegt A, Savelsbergh M (1998) The inventory routing problem. Crainic TG, Laporte G, eds. *Fleet Management and Logistics* (Springer, Boston), 95–113.
- Campbell AM, Wilson JH (2014) Forty years of periodic vehicle routing. *Networks* 63(1):2–15.
- Chen P, Golden B, Wang X, Wasil E (2017) A novel approach to solve the split delivery vehicle routing problem. *Internat. Trans. Oper. Res.* 24(1–2):27–41.
- Choi W, Lee Y (2002) A dynamic part-feeding system for an automotive assembly line. *Comput. Indust. Engrg.* 43(1):123–134.
- Christofides N, Beasley J (1984) The period routing problem. *Networks* 14(2):237–246.
- Da Silveira G, Borenstein D, Fogliatto FS (2001) Mass customization: Literature review and research directions. *Internat. J. Production Econom.* 72(1):1–13.
- De Souza MC, de Carvalho CRV, Brizon WB (2008) Packing items to feed assembly lines. *Eur. J. Oper. Res.* 184(2):480–489.
- Dueck G, Scheuer T (1990) Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *J. Comput. Phys.* 90(1):161–175.
- Emde S (2017) Scheduling the replenishment of just-in-time supermarkets in assembly plants. *OR Spectrum* 39(1):321–345.
- Emde S, Boysen N (2012a) Optimally locating in-house logistics areas to facilitate JIT-supply of mixed-model assembly lines. *Internat. J. Production Econom.* 135(1):393–402.
- Emde S, Boysen N (2012b) Optimally routing and scheduling tow trains for JIT-supply of mixed-model assembly lines. *Eur. J. Oper. Res.* 217(2):287–299.
- Emde S, Gendreau M (2017) Scheduling in-house transport vehicles to feed parts to automotive assembly lines. *Eur. J. Oper. Res.* 260(1):255–267.
- Emde S, Fliedner M, Boysen N (2012) Optimally loading tow trains for just-in-time supply of mixed-model assembly lines. *IIE Trans.* 44(2):121–135.
- Faccio M, Gamberi M, Persona A, Regattieri A, Sgarbossa F (2013) Design and simulation of assembly line feeding systems in the automotive sector using supermarket, kanbans and tow trains: A general framework. *J. Management Control* 24(2):187–208.
- Fathi M, Rodríguez V, Fontes DB, Alvarez MJ (2016) A modified particle swarm optimisation algorithm to solve the part feeding problem at assembly lines. *Internat. J. Production Res.* 54(3): 878–893.
- Francis P, Smilowitz K (2006) Modeling techniques for periodic vehicle routing problems. *Transportation Res. Part B* 40(10):872–884.
- Goeke D, Schneider M (2015) Routing a mixed fleet of electric and conventional vehicles. *Eur. J. Oper. Res.* 245(1):81–99.
- Golz J, Gujjula R, Gunther HO, Rinderer S, Ziegler M (2012) Part feeding at high-variant mixed-model assembly lines. *Flexible Services Manufacturing J.* 24(2):119–141.
- Irnich S, Schneider M, Vigo D (2014) Four variants of the vehicle routing problem. Toth P, Vigo D, eds. *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed. MOS-SIAM Series Optim., Vol. 18 (SIAM, Philadelphia), 241–271.
- Irnich S, Toth P, Vigo D (2014) The family of vehicle routing problems. Toth P, Vigo D, eds. *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed. MOS-SIAM Series Optim., Vol. 18 (SIAM, Philadelphia), 1–33.
- Kilic HS, Durmusoglu MB (2013) A mathematical model and a heuristic approach for periodic material delivery in lean production environment. *Internat. J. Adv. Manufacturing Tech.* 69(5–8):977–992.
- Kovacs AA, Golden BL, Hartl RF, Parragh SN (2014) Vehicle routing problems in which consistency considerations are important: A survey. *Networks* 64(3):192–213.
- Laporte G (2007) What you should know about the vehicle routing problem. *Naval Res. Logist.* 54(8):811–819.

- Larson RC (1988) Transporting sludge to the 106-mile site: An inventory/routing model for fleet sizing and logistics system design. *Transportation Sci.* 22(3):186–198.
- Lian YH, Van Landeghem H (2007) Analysing the effects of lean manufacturing using a value stream mapping-based simulation generator. *Internat. J. Production Res.* 45(13):3037–3058.
- Limère V, Van Landeghem H, Goetschalckx M, Aghezzaf EH, McGinnis LF (2012) Optimising part feeding in the automotive assembly industry: Deciding between kitting and line stocking. *Internat. J. Production Res.* 50(15):4046–4060.
- Monden Y (2011) *Toyota Production System: An Integrated Approach to Just-in-Time*, Fourth ed. (CRC Press, Boca Raton, FL).
- Pisinger D, Ropke S (2010) Large neighborhood search. Gendreau M, Potvin JY, eds. *Handbook of Metaheuristics*, Internat. Series Oper. Res. Management Sci., Vol. 146 (Springer, Boston), 399–419.
- Potvin JY, Kervahut T, Garcia BL, Rousseau JM (1996) The vehicle routing problem with time windows part i: Tabu search. *INFORMS J. Comput.* 8(2):158–164.
- Raa B, Aghezzaf EH (2009) A practical solution approach for the cyclic inventory routing problem. *Eur. J. Oper. Res.* 192(2):429–441.
- Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Sci.* 40(4):455–472.
- Savelsbergh MW (1992) The vehicle routing problem with time windows: Minimizing route duration. *ORSA J. Comput.* 4(2):146–154.
- Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. Maher M, Puget JF, eds. *Principles Practice Constraint Programming—CP98*, Lecture Notes Comput. Sci., Vol. 1520 (Springer, Berlin Heidelberg), 417–431.
- Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* 35(2):254–265.
- Staab T, Klenk E, Galka S, Günther WA (2016) Efficiency in in-plant milk-run systems—The influence of routing strategies on system utilization and process stability. *J. Simulation* 10(2):137–143.
- Toth P, Vigo D, eds. (2014) *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed. MOS-SIAM Series Optim., Vol. 18 (SIAM, Philadelphia).
- Vaidyanathan BS, Matson JO, Miller DM, Matson JE (1999) A capacitated vehicle routing problem for just-in-time delivery. *IIE Trans.* 31(11):1083–1092.
- Webb IR, Larson RC (1995) Period and phase of customer replenishment: A new approach to the strategic inventory/routing problem. *Eur. J. Oper. Res.* 85(1):132–148.
- Zenker M, Emde S, Boysen N (2016) Cyclic inventory routing in a line-shaped network. *Eur. J. Oper. Res.* 250(1):164–178.