Jena Research Papers in Business and Economics

Optimally Routing and Scheduling Tow Trains for JIT-Supply of Mixed-Model Assembly Lines

Simon Emde, Nils Boysen

08/2010

Jenaer Schriften zur Wirtschaftswissenschaft

Working and Discussion Paper Series
School of Economics and Business Administration
Friedrich-Schiller-University Jena

ISSN 1864-3108

Publisher:

Wirtschaftswissenschaftliche Fakultät
Friedrich-Schiller-Universität Jena
Carl-Zeiß-Str. 3, D-07743 Jena

www.jbe.uni-jena.de

Editor:

Prof. Dr. Hans-Walter Lorenz
h.w.lorenz@wiwi.uni-jena.de

Prof. Dr. Armin Scholl
armin.scholl@wiwi.uni-jena.de

www.jbe.uni-jena.de

Optimally Routing and Scheduling Tow Trains for JIT-Supply of Mixed-Model Assembly Lines

Simon Emde, Nils Boysen

Friedrich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Carl-Zeiß-Straße 3, D-07743 Jena, Germany,
{simon.emde,nils.boysen}@uni-jena.de

Abstract
In recent years, more and more automobile producers adopted the supermarket-concept to enable a flexible and reliable Just-in-Time (JIT) part supply of their mixed-model assembly lines. Within this concept, a supermarket is a decentralized in-house logistics area where parts are intermediately stored and then loaded on small tow trains. These tow trains travel across the shop floor on specific routes to make frequent small-lot deliveries which are needed by the stations of the line. To enable a reliable part supply in line with the JIT-principle, the interdependent problems of routing, that is, partitioning stations to be supplied among tow trains, and scheduling, i.e., deciding on the start times of each tow train's tours through its assigned stations, need to be solved. This paper introduces an exact solution procedure which solves both problems simultaneously in polynomial runtime. Additionally, management implications regarding the trade-off between number and capacity of tow trains and in-process inventory near the line are investigated within a comprehensive computational study.

Keywords: Mixed-model assembly lines; Just-in-Time; Material supply; Tow Trains

1 Introduction

With increasing product variety, which nowadays seems inevitable to satisfy highly diversified customer demands, thousands of different parts need to be delivered Just-in-Time (JIT) to a multitude of stations of today's mixed-model assembly lines, e.g., in automobile industry. On the one hand, a reliable and flexible part supply is indispensable,

because otherwise material shortages leading to line-stoppages and hundreds of assembly workers being idle threaten. On the other hand, enlarged safety stocks near the line impede the assembly process within the scarce space of stations. Thus, to enable a reliable, small-lot part supply in line with the JIT-principle an increasing number of automobile producers implements the so-called "supermarket-concept". Here, supermarkets serve as decentralized logistics areas, where parts assembled in neighboring line segments are intermediately stored and sorted according to the needs of the assembly process. The remaining distance between supermarket and assembly line is bridged via small tow trains (or tuggers). A tugger consists of a towing vehicle (driven by an operator), which is connected with a few waggons. These waggons are loaded with parts in a supermarket and, then, a train circulates through assigned stations along its given tour. At each stop, bins filled with parts for the respective station are unloaded and empty bins are returned. Finally, an empty train returns to the supermarket to be reloaded for its next tour.

Clearly, the routing and scheduling of tow trains is an important optimization problem in this context. Each tow train is to be assigned to a subset of stations to be supplied with parts and the sequence of station visits is to be specified (routing). Moreover, for a given route the delivery schedule defining arrivals and departures at each stopover is to be determined (scheduling). Obviously, both problems are heavily interdependent, so that a very complex decision problem arises. However, some real-world conditions in automobile industry, i.e., limited maneuverability of tow trains when driven through the narrow aisles of a shop floor and loading tuggers according to the JIT-principle, allow for some simplifications, so that optimal solutions can be determined in one joint optimization approach. For this purpose, this paper introduces an exact nested dynamic programming procedure with polynomial runtime.

The remainder of the paper is organized as follows. Section 2 characterizes the organizational settings of a supermarket in detail and provides a literature review. Then, Section 3 introduces the routing and scheduling problem of tow trains and formalizes them. The optimization procedure is described in Section 4 and extensions of the base model (and required modifications of solutions procedures) are presented in Section 5. In a comprehensive computational study (Section 6) we investigate the benefits of optimal routes and schedules as well as the elementary trade-off between number and capacity of tuggers and in-process inventory near the line. Finally, Section 7 concludes the paper.

2 Organization principles of JIT-supermarkets and literature review

A supermarket as a decentralized storage area for parts used at nearby line segments substitutes frequent small-lot deliveries for centralized part supply from a remote receiving store, so that the intermediate node "supermarket" can be interpreted as being the in-house logistics counterpart of a cross dock (e.g., see Boysen and Fliedner, 2010). Supermarkets are replenished from receiving store with (comparatively) large industrial trucks, whereas line segments are served with small tow trains. This way, part supply can be adjusted more flexibly to unforeseen events as small-lot deliveries can be quickly

replanned while large-lot deliveries, once made, are hard to revoke. This advantage of the supermarket-concept is very important in today's automobile production as the space at the stations of the line is notoriously scarce. Finally, small-lot deliveries entail smaller bins, which can be stored in comfortable racks near the line. Assembly workers can access parts in an ergonomic and efficient manner, which reduces the strain on the workforce and saves handling time when parts are fetched. The complete part supply process via supermarkets is described in the following:

When an industrial truck arrives at the supermarket, logistics workers sort incoming parts into the racks of the supermarket. There, parts are intermediately stored until a part demand from an assembly station is communicated to the supermarket. Then, a pick list is generated and a logistics worker assembles bins according the pick list, where some parts, e.g., windshields, additionally need to be sorted just-in-sequence as defined by the given assembly sequence of automobiles. Filled bins are loaded on empty waggons and moved to the stopping point of tow trains. Note that, typically, bins are assigned to waggons such that each waggon contains parts for a separate station.

As soon as a towing vehicle arrives at a supermarket's stopping point, the driver couples waggons for the stations assigned to his/her tour and starts visiting them as defined by the tow train schedule. Similar to a bus schedule, it precisely specifies the sequence of station visits and the point in time of each stopover. One automobile producer we visited was experimenting with display panels installed at each station similar to those of bus and railway stations. Here, a countdown until the tugger's next arrival was announced, so that anticipating material shortages in a credible and timely manner got much easier for assembly workers and team leaders.

When a tugger arrives at a station, the driver unloads bins and exchanges empty with filled bins in the material racks of the station without impeding the assembly process. Empty bins are reloaded into the tow train. Some automobile producers have already fully automated these stops by applying "shooter-racks" (see Emde et al., 2009). These special gravity flow racks allow tow train waggons to dock while driving by. As soon as a tugger stops, gates sideways of the waggon and at the back of the rack are opened automatically and bins are injected by elastic springs into the rack and vice versa. These racks reduce the length of a stopover to merely a few seconds, so that reliable tow train schedules can be derived. As soon as all stations on a tugger's tour have been supplied, the vehicle returns to the supermarket, decouples empty waggons and will repeat the above steps to set off on its next tour.

A typical supermarket we observed at multiple German OEMs shows the following properties: A supermarket serves between 20 and 30 stations and is located in direct vicinity of the assigned line segment, so that a complete tow train tour typically amounts to merely 200-500 meters. Three to five tow trains are assigned to each supermarket, so that five stations is a representative number of stopovers visited per tour and up to three visits per station and hour are planned.

In recent years, supermarkets and tow trains became increasingly popular for a JIT-supply of mixed-model assembly lines. However, a "kanban supermarket" is not a novel phenomenon but rather a core element of the famous Toyota Production System (see

Vatalaro and Taylor, 2005, Holweg, 2007) with a long tradition in many industrial sectors (Rees et al., 1989, Hodgson and Wang, 1991, Spencer, 1995).

The planning and control of this in-house logistics concept amounts to a complex task where several interrelated decision problems have to be solved:

(i) Decide on the number and location of decentralized supermarkets and assign line segments.

(ii) Determine the number of tow trains per supermarket and decide on the route of each tugger.

(iii) Determine each tow train's delivery schedule for supplying parts on its given tour.

(iv) Decide on the bins to be loaded per tour of a tow train.

In spite of their great practical relevance, literature on supermarkets and the coordination of tow trains is extremely scarce. Up to now, merely Emde et al. (2009) explicitly treat one of the aforementioned problems. They consider problem (iv) and present an exact solution procedure which determines the bins to be loaded for each tour (of a given schedule) with limited tugger capacity, so that inventory near the line is minimized. Up to now, the other problems have not been considered in the context of supermarkets. However, these in-house logistics decision problems show some similarities to problems of designing and operating traditional distribution networks. This paper jointly treats problems (ii) and (iii), which are related to traditional vehicle routing (see, e.g., Campbell et al., 1998) and inventory routing problems (e.g., Cordeau et al., 2007).

However, the VRP only explicitly covers the routing aspect of the problem and furthermore demands that customers (or stations) are visited exactly once without restrictions on the order in which this is done, which does not reflect the reality of the assembly line parts supply. The IRP on the other hand does take into account multiple deliveries over a longer horizon but presupposes stochastic or constant consumption rates and delivery volumes assigned by the delivery company (or the supermarket) instead of the customers/stations. This runs counter to the JIT-philosophy of most assembly setups where exactly the required amount of parts must be delivered in each tour. The problem considered in this paper therefore falls into neither the VRP nor IRP category.

3 Problem description

For a given line segment of a mixed-model assembly system to be served by a respective supermarket, this paper jointly treats the routing and scheduling problem of tow trains, which is to be solved for each production shift.

The *routing problem* is to determine the fleet size of tow trains to be applied for part replenishment and the partition of the given station set among tuggers. Furthermore, in a typical routing problem the sequence of stopovers has to be decided on. However, two peculiarities in the automobile industry facilitate this additional decision. With multiple waggons attached a tugger cannot drive in reverse direction and, typically, the turning

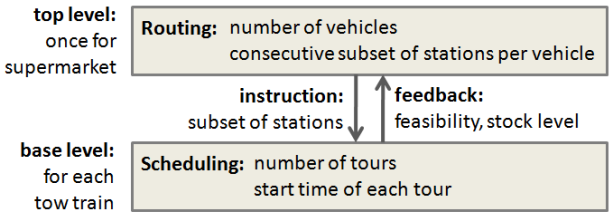


Figure 1: Schematic representation of interdependent routing and scheduling decisions

radius is far too large for the narrow corridors of a shop floor. Thus, merely unidirectional tugger tours along the flow-direction of the assembly line are possible. Furthermore, to avoid congestions of tuggers in narrow aisles, typically automobile producers aim at a consecutive partition of stations among tuggers. This means that it is impossible to have stations 3, 5 and 7 on a route but not the intervening ones. With these two peculiarities on hand, the routing problem reduces to a partition problem of stations into consecutive subsets among a variable number of tow trains. As an extensive fleet of vehicles entails both acquisition as well as maintenance and staffing cost, it is desirable to keep this number minimal. On the other hand, more vehicles facilitate part supply and make more frequent deliveries and thus lower stocks at the line possible.

The *scheduling problem* is to be solved for each tow train and its corresponding subset of stations as determined by the superordinate routing problem. On the one hand, for each tugger its number of tours has to be determined, where a tour comprises a tugger's loading operation in the supermarket and a complete cycle through all its assigned stations. On the other hand, the arrival time of each stopover along each tour is to be specified. However, again special circumstances in automobile industry allow for a simplification of the latter decision task. Due to short distances between supermarket and stations and brief station visits, tours can be approximated as having identical duration irrespective of the bin composition loaded on a tugger. Thus, our scheduling problem reduces to determining the number of tours and the start time of each tour. This decision task aims at avoiding material shortages at the stations while minimizing inventory near the line given the limited capacity of a tugger. Note that less than a handful of waggons can be connected to an engine, because otherwise a tugger cannot be driven safely through the sharp turns of a shop floor. The demand for parts to be delivered is determined by the production sequence of product models launched down the mixed-model assembly line. In the automobile industry, the production sequence is determined three to four days before production starts (and communicated to part suppliers). As the routing and scheduling problem of tow trains is a daily problem solved for each production shift, the production sequence is known with certainty and part demands for each station and production cycle are exactly defined. Clearly, material shortages and resultant line stoppages must be avoided, which, however, is not always possible for any subset of stations as assigned by solving the routing problem and production sequence given the limited tugger capacity. On the other hand, in line with the JIT-principle, inventory is to be reduced. Excessive inventory in a station obstructs assembly operations and moves

reorder dates of previous storage locations forward. Thus, we aim at minimizing the sum of inventories over all production cycles and stations.

Obviously, both problems are heavily *interdependent*. The routing problem determines the subset of stations to be served per tow train. This decision variable of the routing problem serves as an instruction (given parameter) at the subordinate level of scheduling each tugger. On the other hand, only solving the respective scheduling problem can determine whether or not a specific subset of stations allows for a feasible delivery schedule given the limited tugger capacity. Likewise, to exactly determine part inventory at a tugger's assigned stations, a scheduling problem has to be solved. Thus, due to this strong interdependency it seems advisable to solve both problems simultaneously. Figure 1 depicts the decisions tasks of both levels and the relations among them.

To precisely model the joint tow train routing and scheduling problem the following premises are introduced:

- Time units are normalized to the equidistant length of a production cycle, which is not a very restrictive premise as typical cycle times are fairly short, e.g., between 60 and 90 seconds in the automobile industry.
- To avoid congestions in the narrow aisles stations on a tugger's route are always consecutive and served in flow direction of the assembly line.
- All bins are of identical standardized size, which is a requirement of the aforementioned shooter racks. With a given number of waggons per tow train and standardized bins the capacity restriction of vehicles can be measured one-dimensionally by limiting the number K of bins to be loaded. As all tuggers are sufficiently powerful an additional weight restriction is a non-issue.
- Once departed, a tow train will always cycle through all the stations on its route without interruption. The duration of each stopover is not dependent on a tugger's bin load, so that all tours of a single tugger take equal duration.
- In line with the JIT-principle it is assumed that each tugger tour exactly loads the complete number of bins demanded by stations up to the next visit of the tugger. Note that additional bins to avoid capacity bottlenecks on later tours are excluded by this premise. This, however, is a typical policy applied at many OEMs to obviate a sophisticated loading problem (see Emde et al., 2009) to be solved for each tugger.
- W.l.o.g. safety stock must not explicitly be considered, but can simply be added to the first demand period. Furthermore, all items are assumed to be available for consumption in the next production cycle after a tugger's arrival at the respective station.
- The production sequence was previously determined, so that the demand for material bins at each station and in each production cycle is known with certainty. Preprocessing these part demands is clarified by the following example.

Example: Consider the example demand for parts per station and model given in Table 1a. In this problem, there is a production setting with five stations $s = 1, \ldots, 5$ and three different models $k = 1, 2, 3$. For example, assembling one unit of model $k = 3$ will require $d_{13}^{mod} = 2$ parts at station $s = 1$. Note that in this introductory example we restrict ourselves to only one kind of part per station, but extending the problem to account for different parts used in different quantities at different stations is entirely possible.

The total demand for models in the current shift, depending on customer orders, is to be satisfied by the following given production sequence: $< 2, 1, 1, 2, 3]$. With this sequence and the part requirements per model in mind, we can calculate the number of parts used at each station in each cycle. Table 1b shows the numbers of parts per station and cycle: The first unit to be launched down the line is of model 2 which requires one of the parts used at station 1 ($d_{11}^{cyc} = 1$). As the workpieces move down the line sequentially, the unit will arrive in station 2 not before cycle 2 where it consumes 3 parts ($d_{22}^{cyc} = 3$). Meanwhile, in cycle 2, the next unit (of model 1) is launched but does not need any parts in station 1 thus $d_{12}^{cyc} = 0$. The rest of the values are computed analogously.

Parts are usually not delivered one at a time but in bins containing multiple units of one kind of part. While bins are of standard size, parts come in a variety of shapes and sizes therefore differing numbers of parts will fit into a standardized container. The bin capacities in the example can be found in Table 1c. In the table, we can use the station index s to number the parts because, as mentioned before, in this example we have only one kind of part per station.

Now, we can calculate the number of bins needed at each station in each cycle. Consider station 2: Consulting Table 1b, we see that three parts are required in cycle two. As no bins have yet been brought to the station, one bin containing 4 parts must be in stock in this cycle, hence $d_{22} = 1$. Note that the bin is not immediately emptied in cycle 2: A residue of $4 - 3 = 1$ part remains. Production in cycle 3 requires another single part, however this can simply be taken from the bin already at the station, which will then be empty. Cycle 4 claims one more part which is, however, not in stock at the station, thus we must set $d_{24} = 1$. The remaining $4 - 1 = 3$ parts suffice to meet the demand of $d_{25}^{cyc} = 3$ parts in cycle 5 rendering further deliveries unnecessary. The bin demand for all stations and cycles can be found in Table 1d.

d_{sk}^{mod}	1	2	3
1	0	1	2
2	1	3	0
3	3	1	1
4	1	1	0
5	1	0	2

(a) Example demands of parts per station and model.

d_{sc}^{cyc}	1	2	3	4	5	6	7	8	9
1	1	0	0	1	2	0	0	0	0
2	0	3	1	1	3	0	0	0	0
3	0	0	1	3	3	1	1	0	0
4	0	0	0	1	1	1	1	0	0
5	0	0	0	0	0	1	1	0	2

(b) Example demands of parts per station and cycle.

s	size
1	1
2	4
3	4
4	3
5	5

(c) Example bin capacities.

d_{st}	1	2	3	4	5	6	7	8	9
1	1	0	0	1	2	0	0	0	0
2	0	1	0	1	0	0	0	0	0
3	0	0	1	0	1	0	1	0	0
4	0	0	0	1	0	0	1	0	0
5	0	0	0	0	0	1	0	0	0

(d) Example demand of bins per cycle.

Table 1: Example data.

With these premises on hand and the notation summarized in Table 2, a more formalized definition of both problems can be derived.

Routing problem: Given $s = 1, \ldots, S$ stations to be supplied with parts the supermarket routing problem (SRP) aims at a partition of these stations into a variable number $n \in \{1, \ldots, S\}$ of disjunct subsets of consecutive stations each served by a separate tow train. This decision task is encoded by a vector $X(n) = \{0, x_1, \ldots, x_{n-1}, S\} \longrightarrow \{1, \ldots, S-1\}$, where x_i denotes the right hand border of the line segment assigned to tugger i. This vector is of variable length $(n+1)$ and is to be determined such that it minimizes objective function (1) subject to constraints (2):

$$\text{(SRP) minimize } Z(X(n)) = \gamma \cdot n + \sum_{i=1}^{n} F^*(x_{i-1} + 1, x_i) \tag{1}$$

subject to

$$x_i \geq x_{i-1} + 1 \qquad \forall\, i = 1, \ldots, n \tag{2}$$

Objective function (1) aims at minimizing total cost consisting of tugger and inventory costs. Tugger cost comprise, e.g., maintenance and staffing cost per train and, thus, depend on the number n of tow trains applied (weighted with cost factor γ). The additional term of objective function (1) denotes inventory cost resulting from assigning all stations from left border $x_{i-1} + 1$ to right border x_i to tugger i aggregated over all tuggers. These inventory costs per tugger i, however, cannot directly be determined within

S	number of stations (index $s = 1, \ldots, S$)
n	number of tow trains (index $i = 1, \ldots, n$)
C	number of cycles (index $c = 1, \ldots, C$)
m	variable encoding the number of tours on a given route (index $t = 1, \ldots, m$)
γ	total cost of applying a tugger during a shift
δ_s	unit inventory cost at station s per cycle
K	maximum number of containers that can be loaded onto the tow train
$P(l, r)$	number of cycles the tugger needs to reach the supermarket after last station r, be replenished and then reach the first station l of the next tour
$p_{ss'}$	number of cycles the tugger needs to get from station s to station s'
d_{sc}	number of bins in demand at station s in cycle c
a_{ts}	auxiliary variable denoting the number of bins delivered to station s in tour t
x_i	variable encoding the last station on the route of tugger i
y_t	variable encoding the cycle of a tugger's arrival at the first station of tour t

Table 2: Notation

SRP but require the solution of a subordinate scheduling problem, where $F^*(l, r)$ denotes the objective function value of the optimal solution to a tugger's scheduling problem if assigned a line segment ranging from left border l to right border r. Finally, constraints (2) ensure that line segments are consistently ordered from left to right and that each of the n tuggers serves at least one station.

Scheduling problem: For each tugger and its consecutive line segment ranging from station l to r, a supermarket scheduling problem (SSP) is to be solved, which exactly defines the number of tours and their start cycles. The optimal objective value F^* of the SSP is passed back to the SRP as the resulting inventory cost. Note that if no feasible solution for SSP can be determined, i.e., the capacity of a tugger is not sufficient to serve the assigned stations, a prohibitive value (infinity) is returned. Within SSP, start times of a variable number of m tours are encoded as vector $Y(m) = \{-D, y_1, \ldots, y_m, C\} \longrightarrow \{0, \ldots, C - D\}$, where C denotes the overall number of production cycles and $D := p_{lr} + P(l, r)$ the overall duration a tour takes consisting of the duration p_{lr} of cycling through the assigned stations and the replenishment time $P(l, r)$ in the supermarket. The first tour, starting in cycle $-D$, is a "dummy tour" that must not carry any actual load. Inserting such a tour is necessary because otherwise the start cycle y_1 of the real first tour would have to specified in advance. The optimal value for y_1, however, is not an input but the product of the optimization; always setting this to 0, for example, would

not necessarily lead to optimal results. Since a time of at least D cycles must elapse between any two tours, having the dummy tour start in cycle $-D$ will ensure that the true first tour may start in the optimal cycle y_1^* between 0 and the time of occurrence of the first demand at any station on the route. Vector $Y(m)$ is chosen such that it minimizes objective function (3) subject to constraints (4) to (7):

$$\text{(SSP) minimize } F(l,r,Y(m)) = \sum_{t=1}^{m}\sum_{s=l}^{r}\delta_s \cdot \sum_{k=y_t+p_{ls}+1}^{\min\{y_{t+1}+p_{ls};C\}} \left(a_{ts} - \sum_{\tau=y_t+p_{ls}+1}^{k} d_{s\tau} \right) \quad (3)$$

subject to

$$\sum_{k=y_t+p_{ls}+1}^{\min\{y_{t+1}+p_{ls};C\}} d_{sk} = a_{ts} \quad \forall\, t = 0,\ldots,m;\ s = l,\ldots,r \quad (4)$$

$$y_t + p_{lr} + P(l,r) \leq y_{t+1} \quad \forall\, t = 0,\ldots,m \quad (5)$$

$$\sum_{s=l}^{r} a_{ts} \leq K \quad \forall\, t = 1,\ldots,m \quad (6)$$

$$a_{0s} = 0 \quad \forall\, s = l,\ldots,r \quad (7)$$

Equations (4) assign the number of containers delivered in tour t to station s to auxiliary variable a_{ts}. Note that, according to the premises, the tugger brings exactly as many bins to a station as are needed until its next arrival, hence it suffices to add up the demand d_{sk} between two tours t and $t+1$ in each station s in order to get the number a_{ts} of delivered bins. To exactly calculate the point in time parts delivered on tour t are available at station s the time p_{ls} required from start station l to current station s are to be added to start time y_t plus one additional cycle (since parts are available in the next cycle after delivery). Constraints (5) make sure that the tugger has sufficient time to finish a tour before the next one starts, which means that enough time is scheduled for driving through stations (p_{ls}), reaching the supermarket from last station r, reloading the vehicle and arriving at initial station l ($P(l,r)$). Constraints (6) ensure that it is possible to satisfy the demand in between any two tours without exceeding the tugger capacity K while constraints (7) declare the first tour a "dummy tour" without load. The objective function (3) seeks to minimize the sum of all bins lying in stock over all cycles and stations on the route weighted with station specific cost factor δ_s. As the bins that are needed (a_{ts}) in each station s in the interval between any two tours equals exactly the amount delivered, the respective stock in each cycle in that interval can be easily calculated by subtracting the amount of parts consumed up to that cycle from the total demand.

4 Dynamic programming

4.1 Routing

For any tow train, the rightmost station on its route has to be determined; this automatically sets the left border of the next vehicle's route since no station may be left out and only non overlapping routes are allowed (see Figure 2). Therefore, a current tugger's route only depends on its left and right border (and not on the detailed routes of previous tuggers), so that the problem can be evaluated with a dynamic programming procedure.

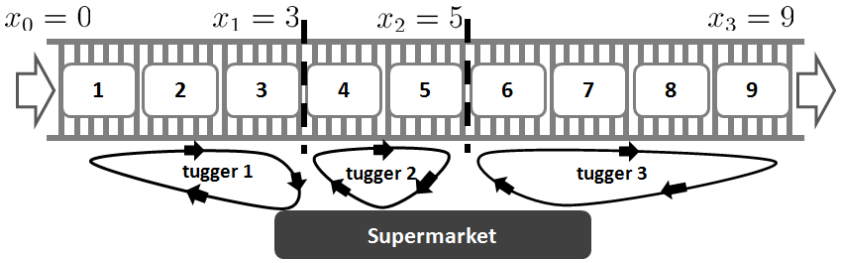


Figure 2: Supermarket supplying nine stations with three tow trains ($i = 1, 2, 3$) on routes defined by left ($x_i + 1$) and right (x_{i+1}) borders representing solution vector $X = \{0, 3, 5, 9\}$.

The decision process is subdivided into $S + 1$ $(0, \ldots, S)$ states, where each state i denotes a station representing the right-hand border of a route. The algorithm operates with a forward recursion. Partial objective values for transitions from state i to another state j, $j > i$, are given by

$$w(i, j) = \gamma + F^*(i + 1, j), \tag{8}$$

where $F^*(i+1, j)$ denotes the objective value of the optimal solution to the corresponding scheduling problem with the given route from station $i+1$ to station j. Let $G(j)$ be the optimal objective value for the station interval from 0 to j and $G(0)$ be zero. Because routes are independent of each other, the following recursion holds:

$$G(j) = \min_{0 \le i \le j-1} \{G(i) + w(i, j)\}. \tag{9}$$

The optimal solution is now given by the path leading to target state S with minimal $G(S)$ and the optimal partition can be determined by a simple backward recovery. A formal definition of the forward recursion is given in Figure 3.

```
1  G(0) := 0;
2  G(j) := ∞        ∀j = 1, ..., S;
3  for j := 1 to S do
4      for i := 0 to j − 1 do
5          if w(i, j) + G(i) < G(j) then
6              p(j) := i;
7              G(j) := w(i, j) + G(i);
8          end
9      end
10 end
```

Figure 3: Dynamic programming algorithm for solving the SRP. Optimal routes will be stored in p, total optimal cost in $G(S)$.

Example (cont.): Figure 4 visualizes the progression of the algorithm for the example problem; vertices denote states/stations and arcs stand for transitions/routes weighted with function w and cost factor $\gamma = 3$. The optimal solution (bold faced) comprises two arcs, i.e., two vehicles are required to optimally service the line in this example, the first one serving stations 1 through 3 and the second stations 4 and 5. This leads to a total objective value of 14. Note that for determining each arc weight w a separate scheduling problem (SSP) is to be solved. Further note that a solution with only one edge does not feasibly exist in the example due to the tugger capacity $K = 10$ being too low to supply all stations with only one vehicle; the arc weight therefore equals ∞.

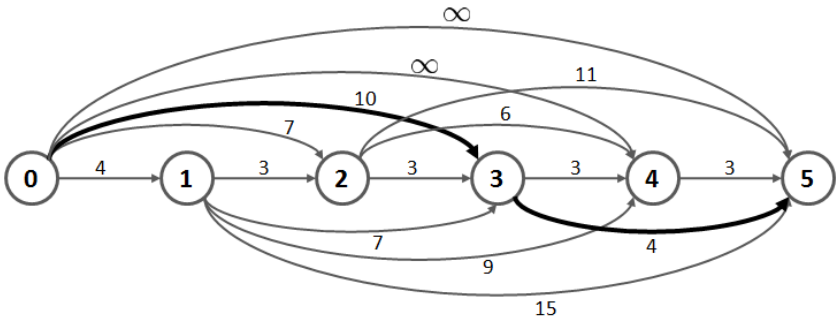


Figure 4: Dynamic programming graph for the example problem; optimal solution is bold.

4.2 Scheduling

In the previous section, we used the optimal objective value F^* to calculate the arc weights $w(l, r)$ in the routing graph, which, of course, must first be determined itself by drawing up the optimal tugger schedule given the route between stations l and r. The tugger, once departed, will take a constant amount of time $D := p_{lr} + P(l, r)$ to cycle through all the stations on its route and to be replenished at the supermarket. During

this time, no further tour can be scheduled, of course, but it can be expedient to delay the next tour for a couple of cycles to reduce inventory. According to the premises, the tugger cannot bring more bins to any station than are needed until its subsequent arrival. The tours are therefore independent of each other. The problem thus consists of dividing an array of C cycles into distinct intervals, each of at least length D, which can also be efficiently solved via dynamic programming.

This problem is best represented by a weighted digraph defined by a three-tuple (V, E, f), with V being the set of nodes, E being the set of arcs between nodes and f the weighting function $f : E \to \mathbb{R}$. Nodes represent cycles in which a tour can end and therefore the next one start, with a unique source node $-D$ and a unique sink C. Keeping in mind that at least D cycles must lie between tours, we can define:

$$V = \{-D\} \cup \{c \,|c = 0, \ldots, C - D\} \cup \{C\} \\ \setminus \left\{ c \left| \sum_{s=l}^{r} \sum_{c'=p_{ls}}^{c+p_{ls}} d_{sc'} > 0; c = 1, \ldots, D-1 \right. \right\},$$

with $d_{sc} = 0 \,\forall\, s = l, \ldots, r;\ c = -D, \ldots, 0$. The first tour starting in $-D$ is a "dummy tour" that must not actually deliver any bins. Therefore this initial tour cannot end in or after a cycle where bins are consumed at a station on the tugger's route. Also, since cycle 0 is the first cycle in which any real tour may start and a tour takes D cycles to complete, a node c, $\forall c = 1, \ldots, D-1$ cannot be the head of an arc if any bins have already been needed before or at c. Such nodes can thus be removed from the graph. Similarly, all cycles in the interval $(C - D, C]$ need not be considered since no tour can start (or end) there because the remaining time is insufficient for completing a tour.

An arc between one node c and another c' exists only if the number of cycles between the two nodes is large enough to accommodate a tour. For reasons already explained, arcs from the dummy source $-D$ can only end in nodes up to which no demand for bins has occurred. Thus, arc set E is defined as follows:

$$E = \left\{ (c, c') \,|c' \geq c + D; c, c' \in V \setminus \{-D\} \right\} \\ \cup \left\{ (-D, c) \left| \sum_{s=l}^{r} \sum_{c'=p_{ls}}^{c+p_{ls}} d_{sc'} = 0; c \in V \setminus \{-D\} \right. \right\}.$$

Keeping in mind that stocks do not carry over from one tour to the next, the weight of an arc from c to c' can be calculated as the sum of the bins lying in stock during the tour interval. When calculating these values, the amount of time, i.e., cycles, the tugger needs to get from one station to the next will also have to be taken into account. If, for example, a tour supplying stations $l = 1$ through $r = 5$ is scheduled to begin in cycle $c = 12$ but the vehicle needs $p_{1,5} = 10$ cycles to even get to station 5, then the amount of stock in station 5 that can be ascribed to the current tour can be only that of cycle 22 onwards. Previous stocks must stem from an earlier tour and must therefore

be added to a preceding arc's weight. Consequently, the weights can be determined with the following function:

$$f(c,c') = \begin{cases} \infty, & \text{if } \sum_{s=l}^{r} a_s > K \\ \sum_{s=l}^{r} \delta_s \cdot \sum_{k=c+p_{ls}+1}^{\min\{c'+p_{ls},C\}} \left(a_s - \sum_{\tau=c+p_{ls}+1}^{k} d_{s\tau} \right), & \text{else} \end{cases} \tag{10}$$

with

$$a_s = \sum_{k=c+p_{ls}+1}^{\min\{c'+p_{ls},C\}} d_{sk} \qquad \forall s = l, \ldots, r.$$

Due to the independence of tours, we can define the following recursion:

$$H(c') = \min_{(c,c') \in E} \left\{ H(c) + f(c,c') \right\}, \tag{11}$$

with $H(-D) := 0$. Figure 5 outlines the corresponding forward recursive dynamic programming algorithm. Note that if $H(C) = \infty$, no feasible schedule with route borders l and r exists.

```
1  Set c_0 to the cycle in which the first tour can start at the latest;
2  D := p_lr + P(l, r);
3  for c := 0 to c_0 − 1 do
4      q(c) := 0;
5      H(c) := 0;
6  end
7  H(c) := ∞ ∀c = c_0, ..., C;
8  for c = 0 to C − D do
9      for c' := c + D to C − D do
10         if H(c) + f(c, c') < H(c') then
11             q(c') := c;
12             H(c') := H(c) + f(c, c');
13         end
14     end
15     if H(c) + f(c, C) < H(C) then
16         q(C) := c;
17         H(C) := H(c) + f(c, C);
18     end
19 end
```

Figure 5: Dynamic programming algorithm for solving the SSP, given the left (l) and right (r) borders of its route. Optimal schedule will be stored in q, total optimal cost in $H(C)$.

Example (cont.): Consider the arc $(1, 3)$ in the routing example above (Figure 4). This arc reads as "one tow train supplies stations 2 and 3", i.e., $l = 2$ and $r = 3$. To calculate the respective arc weight (SRP), the optimal schedule of the vehicle needs to be determined (SSP). First off, the total duration of a tour is $D = p_{2,3} + P(2, 3) = 1 + 2 = 3$ cycles. As such, no adjacent or semi-adjacent nodes can be connected by an arc. Figure 6 depicts the graph for this subproblem in the example. The sink node is $C = 9$, while the source is $-D = -3$; arcs originating from this latter node designate "dummy tours". Looking at Table 1d, we see that in cycle 2 bins are required for the first time in station 2. Since the "dummy tour" must not carry any actual load, its arcs may only connect to nodes 0 and 1, meaning that the cycle when the first real tour starts can be either of those two cycles. In the example we see that it would not make any difference: Both nodes lie on paths with a total length of $H(9) = 4$. One of the optimal solutions (bold in the figure) from -3 to 9 is via 0 and 3. This translates to "the first tour starts in cycle 0, the second in cycle 3", with a total objective value of (given $\delta_1 = \delta_2 = 1$):

$$
\begin{aligned}
F^*(2,3) = G(9) &= f(-3,0) + f(0,3) + f(3,9) \\
&= f(-3,0) + \sum_{s=2}^{3} \delta_s \cdot \sum_{k=1+p_{2s}}^{\min\{3+p_{2s},9\}} \left(\sum_{\tau=1+p_{2s}}^{\min\{3+p_{2s},9\}} d_{s\tau} - \sum_{\tau=1+p_{2s}}^{k} d_{s\tau} \right) + f(3,9) \\
&= \underbrace{0}_{\text{f(-3, 0)}} + \underbrace{1 \cdot ((1-0)+(1-1)+(1-1))}_{\text{station 2}} + \\
&\quad \underbrace{1 \cdot ((1-0)+(1-1)+(1-1))}_{\text{station 3}} + \underbrace{2}_{\text{f(3, 9)}} \\
&= 0 + 2 + 2 = 4.
\end{aligned}
$$

We now have the optimal schedule and the corresponding objective value (inventory cost) for the route supplying stations 2 and 3. Whether or not this route (and hence this schedule) will actually be part of the optimal solution depends on the superordinate routing algorithm. We can see by looking at Figure 4 that in this example the edge $(1,3)$ turns out not to be part of the optimal solution.

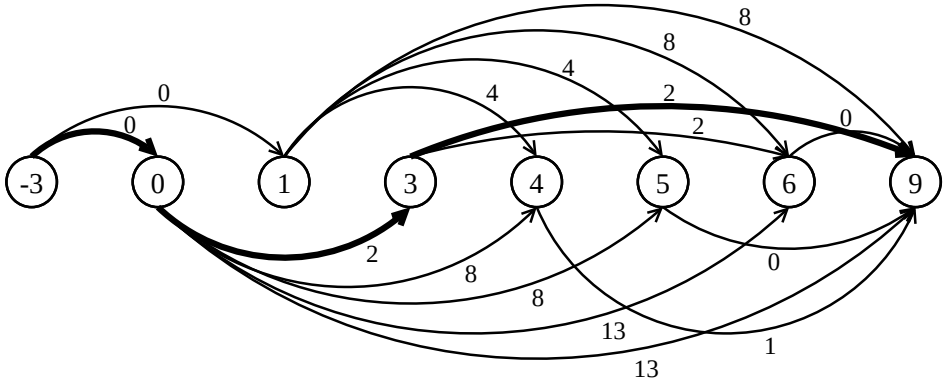


Figure 6: Dynamic programming with $l = 2$ and $r = 3$ in the example; optimal solution is bold.

4.3 Time complexity

The routing dynamic programming (DP) graph for solving the SRP consists of $S + 1$ nodes where each node is connected to at most S others. The number of arcs is thus bounded by $O(S^2)$.

For each arc in the routing graph the scheduling function F^* must be evaluated which necessitates the drawing-up of a tow train schedule. The scheduling DP graph for solving the SSP consists of at most $C+1$ nodes, each node connected to no more than C others. The number of arcs in this graph is therefore bounded by $O(C^2)$.

Finally, the weight f of each arc in the scheduling graph must be determined. With the proposed Equation (10) this can be done in $O(S \cdot C)$ time. However, this function

f can be effectively memoized if the following connection is considered: Let $f^{l,r}(c,c')$ be the weight of the arc from c to c' in the scheduling graph that was created for the route from l to r. If r is raised by one we make the following observation:

$$\begin{aligned} f^{l,r+1}(c,c') &= \sum_{s=l}^{r+1} \delta_s \cdot \sum_{k=c+p_{ls}+1}^{\min\{c'+p_{ls},C\}} \left(a_s - \sum_{\tau=c+p_{ls}+1}^{k} d_{s\tau} \right) \\ &= f^{l,r}(c,c') + \delta_{r+1} \cdot \sum_{k=c+p_{l,r+1}+1}^{\min\{c'+p_{l,r+1},C\}} \left(a_{r+1} - \sum_{\tau=c+p_{l,r+1}+1}^{k} d_{r+1,\tau} \right) \end{aligned}$$

with

$$a_{r+1} = \sum_{k=c+p_{l,r+1}+1}^{\min\{c'+p_{l,r+1},C\}} d_{r+1,k}$$

Storing the $f^{l,r}(c,c')$ values, the need for summing over all stations is eliminated. The arc weights can thus be calculated in $O(C)$. Note that, for simplicity's sake, we omitted the feasibility check in the above equation; the sum $a^{l,r}(c,c') = \sum_{s=l}^{r} \sum_{k=c+p_{ls}+1}^{\min\{c'+p_{ls},C\}} d_{sk}$ needs to be likewise memorized. This memoization technique leads to a space requirement bounded in $O(S^2 \cdot C^2)$ and results in a total time complexity for the whole algorithm of $O(S^2 \cdot C^2 \cdot C)$. Assuming that the number of cycles C is greater than the number of stations S, which is very reasonable in practice, this equates to $O(C^5)$.

5 Modifications of basic tow train routing and scheduling

In this section several extensions and modifications of both the routing and scheduling problems and algorithms are investigated. First, we will discuss some simplifications of the original problems often encountered in practical applications and then we will look at ways to avoid the potentially difficult-to-determine inventory cost δ_s in objective function (3).

5.1 Cyclic deliveries

To make part supply of tow trains even more reliable and easier to supervise some automobile producers aim at cyclic deliveries, which means that a tour of a specific tugger starts in equidistant time intervals, e.g., every twentieth cycle. Then, each assembly worker knows the (planned) arrival of the tugger's next visit and potential material shortages can be anticipated earlier. However, in this case, the question of the optimal interval arises. Figure 7 outlines a simple algorithm that can be used to create a cyclic delivery schedule. The algorithm simply tests all possible combinations of start cycles, the number of which is bounded by C, and numbers of tours, also bounded by C, each of which entails the evaluation of $\lfloor C/D \rfloor$ (bounded by C) f values, which can be done in $O(C)$ time each as we have seen, leading to a time complexity bounded in $O(C^4)$. Using

these F values to calculate the arc weights in the routing DP, the total time complexity of the whole algorithm (routing + scheduling) rises to $O(C^6)$.

```
1  Set c_0 to the cycle in which the first tour can start at the latest;
2  F* := ∞;
3  for c := 0 to c_0 − 1 do
4      for t := 1 to ⌊(C − c)/D⌋ do
5          F := 0;
6          for i := 1 to t do
7              F := F + f(c + ⌈(i − 1) · (C − c)/t⌉, c + ⌈i · (C − c)/t⌉);
8          end
9          if F < F* then
10             F* := F;
11             t* := t;
12             c* := c;
13         end
14     end
15 end
```

Figure 7: Algorithm for constructing cyclic delivery schedules, given a route. Best number of tours will be stored in t^*, best start cycle of first tour y_1 will be c^* and corresponding objective value will be F^*.

5.2 Equal-length routes

For reasons of fairness and to omit a sophisticated routing problem car manufacturers might aim at dividing the stations to be served equally among tuggers and their respective operators. A suitable algorithm is outlined in Figure 8: For every possible number of tow trains $n = 1, \ldots, S$, stations are divided equally among the n vehicles and objective values $Z(X(n))$ are calculated. The vehicle count n^* with the lowest $Z(X(n^*))$ is returned. Note that in order to evaluate Z the subordinate scheduling problem will still have to be solved. This can either be done optimally with the DP or simply by assigning cyclic delivery schedules with the algorithm from the preceding section. Either way, the complexity of the routing itself (sans scheduling) remains obviously bounded by $O(S^2)$.

```
1  for n := 1 to S do
2      segment := S/n;
3      x_0 := 0;
4      x_n := S;
5      for i := 1 to n − 1 do
6          x_i := ⌈segment · i⌉
7      end
8      Calculate Z(X(n));
9      Store the lowest Z =: Z* and the corresponding X =: X*;
10 end
```

Figure 8: Algorithm for assigning equal-length routes. Best solution will be stored in X^* and corresponding objective value in Z^*.

5.3 Fleet size vs. inventory

While a trade-off between the number of tow trains and associated capital consumption, staffing and maintenance cost on the one hand, and the amount of in-process inventory required at the line on the other most certainly exists, it may be difficult to valuate these two very different cost factors in one joint objective function. Especially, unit inventory cost might only be imputed cost for impeding the assembly process, e.g., by initiating additional searching and sorting effort for assembly workers. Also, the effect of moving reorder dates of storage locations forward, when prematurely moving inventory from supermarket to the line, might be hard to quantify accurately over the multiple stages relevant in the real-world, e.g., station, supermarket, central stock, supplier. One way to avoid this quantification problem is to split the two metrics up and construct an efficient frontier, plotting them against each other. The decision maker can then easily see the additional benefit in terms of reduced inventory a greater tow train fleet would afford him and may thus decide whether it is worth it or not.

This can be achieved by removing the number of vehicles n from the objective function (i.e., setting $\gamma := 0$) and fixing it for the DP algorithm. The number of tow trains in a solution is determined by its number of arcs in the DP graph, therefore if, for example, $n = 2$ is given, only paths from source 0 to sink S with two edges may be considered. The shortest such path is then the optimal solution for the given n. If this is done for all $n = 1, \ldots, S$, the efficient frontier is the result.

Example (cont.): Figure 9 shows the graph in the example for $n = 2$. The grey arcs were inserted in the first step and denote all (partial) solutions with $n = 1$. Building on these, the black arrows represent all (partial) solutions with $n = 2$; black arcs with head node 5 are complete, feasible solutions with two vehicles. Of these, the lowest-cost solution is bold in the figure, resulting in an objective value of 8. Note that it would be an easy matter to add another set of arcs continuing the path of the black ones and thus obtain the optimal solution for $n = 3$ and so on.

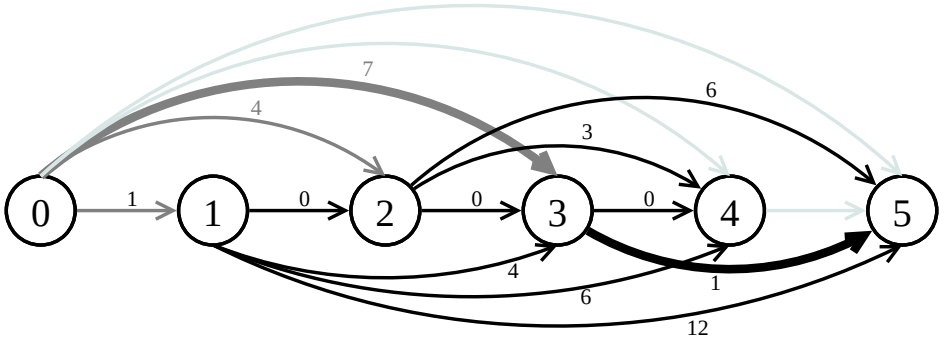


Figure 9: Dynamic programming with given number of vehicles $n = 2$. Optimal solution is bold.

Table 3 contains the optimal objective function values Z^* with $\gamma = 0$, i.e., the in-process inventory, for all possible numbers of tow trains $n = 1, \ldots, 5$. As seen earlier, the optimal solution to this example problem with $\gamma = 3$ made use of two tuggers, while a look at the table below reveals that the sum of all inventory at the line can be reduced by another 37.5% if an extra tugger is added. Whether this decrease in inventory outweighs the additional cost of increasing the size of the tow train fleet is for the decision maker to ascertain.

n	1	2	3	4	5
Z^*	infeasible	8	5	2	1

Table 3: Optimal objective function values Z^* ($\gamma = 0$) for $n = 1, \ldots, 5$ tuggers.

The modified DP algorithm for constructing the entire efficient frontier has to solve the routing problem n times, where n is obviously bounded in S. The time complexity of the routing is hence bounded in $O(S^3)$. Note that, alternatively, the complete routing graph can be constructed upfront (see Section 4.1) and, then, the shortest path procedure of Saigal (1968) can be applied to determine all shortest paths with $i = 1, \ldots, n$ steps. However, the number of scheduling problems to be solved remains the same (i.e., bounded by $O(S^2)$) because the objective function $F^*(l, r)$ can obviously supply at most S^2 unique values seeing that both its parameters l, r are in $\{1, \ldots, S\}$. Solving these scheduling problems before starting the routing and storing the results, the runtime of the whole algorithm remains bounded by the complexity of the scheduling algorithm and thus unchanged, assuming that $C > S$. The space requirements for saving the scheduling results are bounded in $O(S^2)$. Table 4 summarizes all combinations of routing and scheduling algorithms presented in this paper and their aggregated time complexity.

		routing		
		DP given γ, δ	DP, eff. frontier	equal-length routes
scheduling	DP	$O(C^5)^*$	$O(C^5)^*$	$O(C^5)$
	Cyclic deliveries	$O(C^6)$	$O(C^6)$	$O(C^6)$

Table 4: Time complexity for all presented algorithms (assuming $C > S$) with routing and scheduling combined; combinations that solve both SRP and SSP to optimality are marked with an asterisk.

6 Computational study

6.1 Instance generation

As there is no established test data for the joint tow train routing and scheduling problem (SRP/SSP), we will first describe how the instances for this paper were generated.

We aim at instances being of representative size for real-world automobile assembly. Thus, we presuppose 400 production cycles, which is a typical output per shift, and up to 60 stations served by the supermarket. With these parameters on hand SRP/SSP instances are derived from the parts usages of different models at the stations. Depending on the production sequence of the models, the number of parts and, consequently, containers consumed at any station in between any two tours will fluctuate. For each station count from Table 5, the models are randomly generated by assigning a demand d^{par}_{mw} for parts $w \in W$ to each model $m \in M$. The parts usages d^{par}_{mw} are calculated as $\lfloor rnd(u_m, u_m) \rceil$, $\forall m \in M, w \in W$, where $u_m = rnd(0.5, 0.5)\ \forall m \in M$ and $rnd(\mu, \sigma) \sim N(\mu, \sigma)$ is a normally distributed random number greater than 0 and $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. Bins have differing capacities depending on what kind of part $w \in W$ is stored in them, namely a uniformly distributed random number from the interval $[1; 20]$, to allow for the fact that different parts may have different sizes while the size of the bins is standardized. At each station three different kinds of parts w are used. For each station count from the table, 50 random sequences of 400 units each were created. Each of these was tested with varying tugger capacities K and delays between tours P as printed in the table, so that in total 5,400 instances were generated. Travel times P from/to the supermarket as well as between stations p are set to constant values as given below. The cost factor δ (unit inventory cost) is normalized to 1 for all stations.

Symbol	description	values
$\|M\|$	number of distinct models	100
C	number of production cycles / units per sequence	400
K	capacity of each tugger	5, 10, 15, 20, 25, 30
P	delay between tours	1, 5, 10
p	driving time between stations	1
S	number of stations	10, 20, 30, 40, 50, 60

Table 5: Parameters for instance generation

6.2 Computational results

The two nested DP algorithms were implemented in C# 2008 and run on an x86 PC with an Intel Core 2 Quad Q9550 2.8 GHz CPU and 4096 MB of RAM. To avoid quantifying tugger and inventory cost in a joint cost function (see Section 5.3), we will use the DP algorithms to construct efficient frontiers in this computational evaluation and thus regard only the inventory at the line as optimal objective value Z^* (i.e., $\gamma = 0$). The scheduling problems were solved before the routing started and the results stored. As these schedules are independent of each other, they can be calculated in parallel on a multi-core/multi-CPU workstation. Corresponding CPU times ordered by station count S, averaged over all 50 sequences per parameter set, can be found in Table 6, listing both the single-threaded and four-threaded runtimes. Keeping in mind the sequence length of $C = 400$ and the fact that the algorithm outputs both optimal routes as well as optimal schedules for all possible numbers of vehicles $n = 1, \ldots, S$ in just one go, running times of under 10 seconds even in the instances with $S = 60$ stations make the algorithm appear well-suited to solving even the most difficult real-world instances in good time, especially considering the remarkable speed-up achievable through parallel processing, by about a factor of three on a four-core CPU.

S	10	20	30	40	50	60
CPU time ST	1	4	5	10	15	26
CPU time MT	<1	2	2	4	5	9

Table 6: CPU time of the proposed algorithm in seconds, both on a single thread (ST) and on four threads (MT); $C = 400$, $K = 20$ and $P = 5$.

Table 7 shows the average results for the 50 sequences with $S = 10$ stations and a replenishment time of $P = 5$ cycles. Remember that the algorithm outputs optimal routes and schedules for all possible numbers of vehicles, in this case $n = 1, \ldots, 10$

(the rows of the table). The decision maker can thus judge if the improvement in the objective function value (labeled Z^* in the table) is sufficient to offset the additional cost another tow train at the supermarket would infer. Apart from the optimal Z^* values obtained by our nested dynamic programming algorithms, for comparison Table 7 also lists the average objective function values for solutions with unoptimized (equal-length) routes with optimal schedules (the column labeled Eq.-len. routes), optimal routes with unoptimized schedules (Cyclic deliveries) and both unoptimized (Both non-DP) with the procedures as described in the preceding section. Values in brackets denote the number of sequences (out of the total of fifty) for which a feasible solution could be found, i.e., where the line did not starve for parts at any time.

Looking at the table, the first thing that comes to attention is the lack of feasible solutions when only one tow train is available. Even with the maximum capacity of $K = 25$ and optimal routes and schedules, it is not possible to supply all ten stations with just one vehicle in this data set. Differences between the various algorithms first become obvious when considering the solutions with $n = 2$ tuggers: With a capacity of $K = 15$ the line could be feasibly supplied in 30% of the sequences, provided that both routes and schedules are optimal. Concerning objective values, there is a consistent trend discernible with all the instances: Optimal schedules are more important than optimal routes when it comes to reducing surplus stock at the line. Consequently, simply using equal-length routes turns out to be a fair approximation of optimal routing, whereas cyclic delivery schedules tend to be significantly worse than optimal ones. If both routes and schedules are naively (non-optimally) set, tugger capacities need to be adjusted to achieve feasibility and even then objective function values are about 69% greater than optimal ones on average. Optimal routes do not improve these figures much as long as the schedules remain non-optimal (about 65% worse objective values) while optimal schedules help a lot even if the routes are non-DP-assigned (only about 3% worse than optimal). These observations are visualized in Figure 10b for $S = 50$.

This graph also illustrates that, while using more vehicles to supply the stations always improves objective function values, no such thing can be said for the tugger capacity K. The latter parameter is only useful to at all reach feasibility. Once achieved, further increasing K has little to no effect, regardless of the routing and scheduling method used, which can also be seen in Figure 10a. This graph also shows the influence of the number of tow trains n on Z^*: Although more vehicles entail better solutions (i.e., reduced surplus stock), this additional benefit is marginalized with increasing n. In the figure, the graph more or less flattens out after $n = 15$.

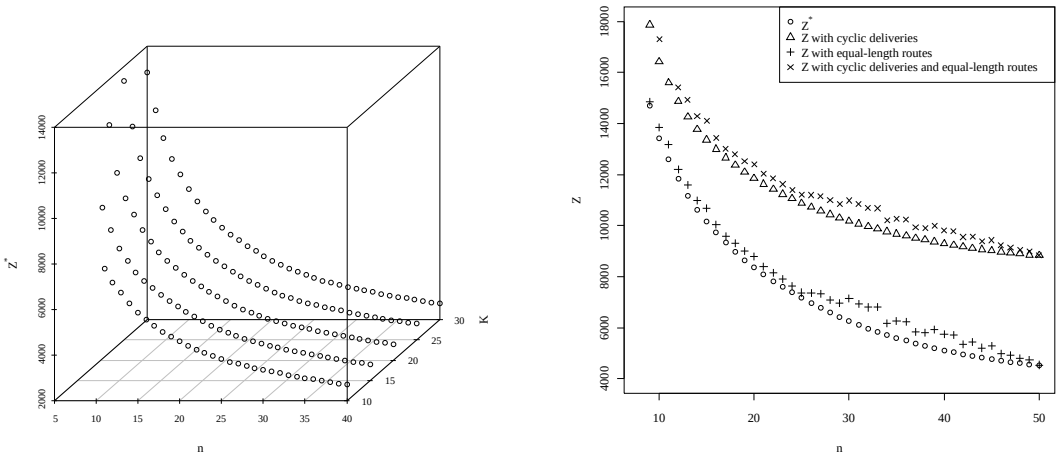
n	K	Cost (# feasible)	Eq.-len. routes (# f.)	Cyclic deliveries (# f.)	Both non-DP (# f.)
	5	- (0)	- (0)	- (0)	- (0)
	10	- (0)	- (0)	- (0)	- (0)
1	15	- (0)	- (0)	- (0)	- (0)
	20	- (0)	- (0)	- (0)	- (0)
	25	- (0)	- (0)	- (0)	- (0)
	5	- (0)	- (0)	- (0)	- (0)
	10	- (0)	- (0)	- (0)	- (0)
2	15	2.326,20 (15)	2.420,40 (5)	2.914,14 (7)	- (0)
	20	2.272,49 (47)	2.314,59 (39)	2.786,39 (46)	2.864,08 (37)
	25	2.261,42 (50)	2.307,29 (49)	2.758,60 (50)	2.827,12 (49)
	5	- (0)	- (0)	- (0)	- (0)
	10	1.770,39 (18)	1.796,67 (3)	2.426,63 (8)	- (0)
3	15	1.701,28 (50)	1.711,36 (47)	2.274,26 (50)	2.318,04 (45)
	20	1.697,68 (50)	1.709,82 (50)	2.262,00 (50)	2.304,44 (50)
	25	1.697,56 (50)	1.709,70 (50)	2.262,00 (50)	2.304,44 (50)
	5	- (0)	- (0)	- (0)	- (0)
	10	1.373,47 (49)	1.396,65 (43)	2.016,90 (48)	2.075,88 (34)
4	15	1.367,08 (50)	1.393,90 (50)	1.992,22 (50)	2.055,13 (48)
	20	1.367,08 (50)	1.393,90 (50)	1.992,22 (50)	2.056,00 (50)
	25	1.367,08 (50)	1.393,90 (50)	1.992,22 (50)	2.056,00 (50)
	5	- (0)	- (0)	- (0)	- (0)
	10	1.149,14 (49)	1.191,70 (43)	1.833,10 (48)	1.911,03 (35)
5	15	1.148,36 (50)	1.188,28 (50)	1.825,98 (50)	1.889,56 (48)
	20	1.148,36 (50)	1.188,28 (50)	1.825,98 (50)	1.889,48 (50)
	25	1.148,36 (50)	1.188,28 (50)	1.825,98 (50)	1.889,48 (50)
	5	1.242,00 (1)	- (0)	- (0)	- (0)
	10	997,94 (49)	1.097,90 (48)	1.720,71 (48)	1.804,34 (47)
6	15	997,26 (50)	1.096,68 (50)	1.715,90 (50)	1.799,94 (50)
	20	997,26 (50)	1.096,68 (50)	1.715,90 (50)	1.799,94 (50)
	25	997,26 (50)	1.096,68 (50)	1.715,90 (50)	1.799,94 (50)
	5	1.017,00 (1)	- (0)	- (0)	- (0)
	10	892,24 (49)	922,92 (49)	1.642,60 (48)	1.679,56 (48)
7	15	891,88 (50)	922,42 (50)	1.637,98 (50)	1.674,72 (50)
	20	891,88 (50)	922,42 (50)	1.637,98 (50)	1.674,72 (50)
	25	891,88 (50)	922,42 (50)	1.637,98 (50)	1.674,72 (50)
	5	891,00 (1)	- (0)	- (0)	- (0)
	10	801,47 (49)	826,47 (49)	1.580,54 (48)	1.607,35 (48)
8	15	801,34 (50)	826,08 (50)	1.576,04 (50)	1.603,38 (50)
	20	801,34 (50)	826,08 (50)	1.576,04 (50)	1.603,38 (50)
	25	801,34 (50)	826,08 (50)	1.576,04 (50)	1.603,38 (50)
	5	813,00 (1)	- (0)	- (0)	- (0)
	10	726,08 (49)	766,65 (49)	1.529,96 (48)	1.561,17 (48)
9	15	726,00 (50)	766,36 (50)	1.525,70 (50)	1.556,86 (50)
	20	726,00 (50)	766,36 (50)	1.525,70 (50)	1.556,86 (50)
	25	726,00 (50)	766,36 (50)	1.525,70 (50)	1.556,86 (50)
	5	771,00 (1)	771,00 (1)	- (0)	- (0)
	10	674,94 (49)	674,94 (49)	1.493,98 (48)	1.493,98 (48)
10	15	674,90 (50)	674,90 (50)	1.489,84 (50)	1.489,84 (50)
	20	674,90 (50)	674,90 (50)	1.489,84 (50)	1.489,84 (50)
	25	674,90 (50)	674,90 (50)	1.489,84 (50)	1.489,84 (50)

Table 7: Results for $S = 10$.

In Figure 11a the following question is explored: What effect does the length of the line segment served by a single supermarket have on the stock kept at the stations? In other words, given a number of tow trains per supermarket, how many supermarkets should be created along the line? The graph shows that the relationship between the number

of stations served by one supermarket and Z^* is almost linear, meaning that, ceteris paribus, doubling the stations served by a single supermarket also doubles the number of bins in stock at each station on average. On the other hand, space constraints and cost caused by creating and maintaining additional supermarkets will also play a role in deciding on the optimal number of supermarkets installed per assembly line. Knowing about the linear connection between station count and stock at the line may, however, be helpful in determining the optimal trade-off.

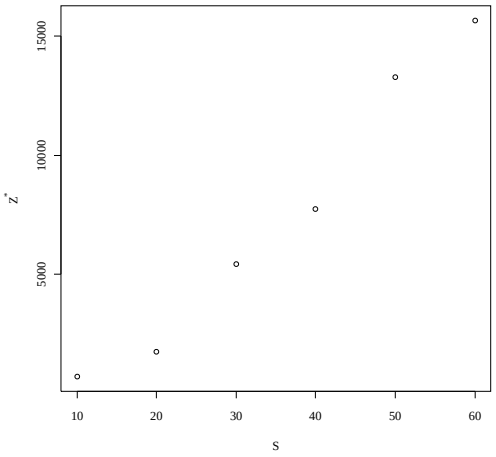
Finally, Figure 11b shows the effects of the replenishment time P on Z^*. Recall that P may be interpreted as the resupply time that the tugger has to wait out at the supermarket and as the transit time between supermarket and line segment. The graphs clearly show that increasing replenishment times have a considerable adverse effect on the objective. The individual values ($P = 1, 5, 10$ cycles in the figure) may not be particularly important, seeing that these parameters cannot be changed on short notice in practice anyhow, however the fact that the time of turnover so distinctly increases the necessary stock at the line is a clear vindication of the supermarket concept as and of itself: It is the very purpose of supermarkets to shorten resupply times and facilitate quick deliveries to the line. In other words: Supermarkets have a naturally low P, which quite obviously pays.



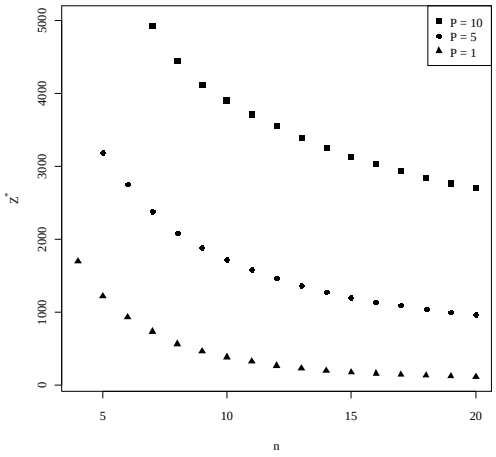
(a) Effect of vehicle number n and capacity K on objective function values.

(b) Difference between simple and optimal solutions.

Figure 10: Effects of vehicle number and capacity as well as delivery policy on Z^*.



(a) Effect of station number S on objective function values with $n = 10$, $K = 30$ and $P = 5$.



(b) Effect of vehicle number n and replenishment time P on objective function values with $S = 20$ and $K = 10$.

Figure 11: Effects of the number of stations and replenishment time on Z^*.

7 Conclusion

This paper studies the joint tow train routing and scheduling problem, which is to be solved by automobile manufacturers supplying the stations of their mixed-model assembly lines via decentralized in-house logistics areas, labeled supermarkets. Being loaded in these supermarkets, small tow trains travel along their assigned line segments and supply parts assembled at the stations according to the JIT-principle. The routing and scheduling problem partitions the stations among tuggers and decides on the start times of each tugger's tours. A nested dynamic programming procedure for solving this problem to optimality in polynomial runtime is presented and in a comprehensive computational study the trade-off between the number of tuggers and their capacity on the one hand and inventory held at the stations on the other is investigated.

Future research on supermarkets and tow trains should especially treat the not yet considered decision problem (see Section 2) of determining the number and location of supermarkets to be installed for supplying a complete assembly line. Additionally, the benefit of the supermarket concept should be compared to alternative forms of organizing part supply, e.g., more centralized alternatives. This way decision support for the practitioner could be gained to decide on the right kind of part supply for his/her specific problem setting.

References

Boysen, N.; Fliedner, M. (2010): Cross Dock Scheduling: Classification, Literature Review and Research Agenda. Omega 38, 413–422.

Campbell, A.; Clarke, L.; Kleywegt, A.; Savelsbergh, M.W.P. (1998): The Inventory Routing Problem. In: Crainic, T.G.; Laporte, G. (eds.): Fleet Management and Logistics. Kluwer Academic Publishers, Dordrecht, Netherlands, 95–113.

Cordeau, J.-F.; Laporte, G.; Savelsbergh, M.W.P.; Vigo, D. (2007): Vehicle Routing. In: Barnhart, C.; Laporte, G. (eds.): Handbook in Operations Research & Management Science 14, Elsevier, Amsterdam, Netherlands, 367–428.

Emde, S.; Fliedner, M.; Boysen, N. (2009): Optimally loading clocked tow trains for JIT-supply of mixed-model assembly lines. Jena Research Papers in Business and Economics (JBE) 10/2009, FSU Jena, Germany.

Hodgson, T.J.; Wang, D. (1991): Optimal hybrid push/pull control strategies for a parallel multistage system: part I. International Journal of Production Research 29, 1279–1287.

Holweg, M. (2007): The genealogy of lean production. Journal of Operations Management 25, 420–437.

Rees, L.P.; Huang, P.Y.; Taylor III, B.W. (1989): A comparative analysis of an MRP lot-for-lot system and a Kanban system for a multistage production operation. International Journal of Production Research 27, 1427–1443.

Saigal, R. (1968): A constrained shortest route problem. Operations Research 16, 205–209.

Spencer, M. (1995): Production planning in a MRP/JIT repetitive manufacturing environment. Production Planning & Control 6, 176–184.

Vatalaro, J.; Taylor, R. (2005): Implementing a Mixed Model Kanban System: The Lean Replenishment Technique for Pull Production. Productivity Press, Portland, OR.