
Computational Optimization and Applications, 31, 295–308, 2005

Shunting Minimal Rail Car Allocation

MARCO E. LÜBBECKE m.luebbecke@math.tu-berlin.de
Technische Universität Berlin, Institut für Mathematik, Sekr. MA 6-1, Straße d. 17. Juni 136, D-10623 Berlin, Germany

UWE T. ZIMMERMANN u.zimmermann@tu-bs.de
Institute of Mathematical Optimization, Braunschweig University of Technology, Pockelsstraße 14, D-38106 Braunschweig, Germany

Received June 23, 2003; Revised July 27, 2004; Accepted September 3, 2004

Abstract. We consider the rail car management at industrial in-plant railroads. Demands for loaded or empty cars are characterized by a track, a car type, and the desired quantity. If available, we assign cars from the stock, possibly substituting types, otherwise we rent additional cars. Transportation requests are fulfilled as a short sequence of pieces of work, the so-called blocks. Their design at a minimal total transportation cost is the planning task considered in this paper. It decomposes into the rough distribution of cars among regions, and the $\mathcal{NP}$-hard shunting minimal allocation of cars per region. We present mixed integer programming formulations for the two problem levels. Our computational experience from practical data encourages an installation in practice.

Keywords: Mixed integer programming, shunting minimization, decomposition, rail transport

MSC (2000): 90C11, 90C27, 90B06

1. Rail car management at in-plant railroads

In-plant railroading is an indispensable mode of freight transport in the industrial sector. Loaded and empty rail cars enter an industrial plant; appropriate cars are then intermediately stored and distributed among terminals; shipments and surplus empty cars leave the plant again. We essentially deal with the problem of which cars to move, where and how. In contrast to public transport, in-plant railroad service is explicitly and only on demand. The railroad's customers, e.g., a production terminal, issue a *transportation request* which specifies a track, a type of cargo (for an unloading terminal) or a car type (for a loading terminal), the tonnage or the number of cars needed, and possibly a delivery time window. The requested car type may be substituted by similar types. In allocating individual rail cars to requests, operational goals are (a) little shunting efforts, (b) short transportation times, and (c) small car rental fees as explained below.

Some special car types serve a single purpose only and are almost always assigned a particular terminal. Aside from these, we are free to allocate practically *any* car of appropriate type. In practice restrictions apply, of course, in order to avoid excessive shunting, spare a locomotive, shorten travel distances and the like. However, we are not limited to use physically present cars, but may assign also those which are known to arrive at the plant

shortly, or e.g., cars soon to return from maintenance. Occasionally, a request is deferred e.g., when its immediate service would unduly consume resources. In the case that unsatisfied demand remains, additional cars must be rented from other railroads. This is part of the normal operation, since cars leave the plant regularly, but incurs a rental cost.

In practice, this process currently results in renting cars for a particular request, and returning the cars upon its completion. In contrast, we allow for re-assigning emptied cars to further requests. Additional issues include that cars may be available only subject to certain conditions e.g., in winter; cars should be assigned preferably starting from the head of a track; there may be priorities of terminals; and there are forbidden follow-up requests. As an example for the latter consider cars in the chemical industry which may or may not need cleaning before they can be assigned to the next request. This depends on the previously transported chemicals. It is now crucial that transportation requests are served as a sequence of up to three pieces of work, called *blocks*. A typical example is (a) to make up a train of cars originating from several tracks, (b) haul the train to a different area of the plant, and (c) distribute the cars to their final destinations. Blocks are the smallest non-interrupted tasks assigned to a locomotive. Consecutive blocks may very well be performed with pauses in between, even on different locomotives. This paper is about the optimal design of such blocks. Given a list of transportation request, we have to come up with a selection a cars of appropriate type which together fulfill the total demand, and a proposal about which cars to group into which blocks.

The current in-plant planning is mostly on a first come first served basis. Software support is only administrative; all managerial decisions rely on skills and experience of the planner. The present research serves as the algorithmic basis for a computer aided planning tool. Its aim is to provide an active proposal for the allocation of cars, thus relieving the planner at the very least from routine tasks.

Related work

Discrete optimization models have been proposed for several planning tasks in rail freight transport, three of which are of some relevance to our situation: empty car distribution, railroad blocking, and shunting. Note that we are not dealing with scheduling the locomotives; we refer to our previous work [11] in this context.

The first problem is about relocating empty rail cars over comparatively long distances of a rail network in order to compensate for geographical imbalances in the demand, see the reviews [3, 14]. Model formulations on time-expanded networks involve the classical transportation problem [1] and integer multicommodity flow problems, where commodities correspond to car types [8, 9, 13]. Models impose capacity constraints on trains, respect train schedules, and allow for shortage of cars. In contrast, we have to access rail cars individually, and our model has to reflect this more detailed resolution of the data.

Rail cars originating from different stations usually share pieces of their routes until they reach their possibly different destinations. To this end, cars are reclassified in stations on their way. A group of cars with a common (intermediate) origin-destination pair is usually called a *block*, this is where we lend our terminology from. Railroad blocking, that is, assigning each car a sequence of blocks is a major problem in railroad scheduling, and

received some attention in the literature, see e.g., [2, 12]. Proposed models are able to exactly solve problems with thousands of shipments. However, blocking in our situation is much easier since the eligible combinations of sequences of blocks are much more restricted due to shorter distances and fewer simultaneous shipments.

A shunting problem in rail yards discussed in [5] deals with regrouping trains according to a given sequence, using as few additional tracks as possible. In contrast, we only have to access a certain number of cars of given types, and aim at a selection of cars which incurs the smallest shunting efforts. On-line and real-time aspects of shunting street cars are considered in [15].

Except from an aged paper [4] the problems particular to in-plant railroads as a *combination* of the above three aspects, have not been mentioned, let alone solved in the operations research literature.

2. Decomposition into regions

A very natural approach to our problem is suggested by the current planning practice: A decomposition of the railroad track network into *regions*. Groups of neighboring tracks frequently serve a common purpose, e.g., as storage areas or as tracks dedicated to incoming trains. Each track belongs to a unique region. A block must not contain cars originating from or destined to different regions. In other words, a block starts and ends in the same region, or it is a direct connection between two regions. That is, our rail car allocation problem decomposes in two subproblems: The rough distribution of cars among regions at an upper level, and the allocation of cars per region at a lower level.

In this section we are concerned with the upper level. Given a set R of transportation requests, let us denote by $\mathcal{T}_r$ the set of car types admissible for request $r \in R$. The network is split into disjoint regions $i = 1, \ldots, n$, for each of which we know its supply $a_i^\tau \geq 0$ of the respective car type τ. Further, for each $r \in R$, denote by $b_r \geq 0$ its demand, as a number of cars. For each τ, we introduce a super source S_τ which represents renting at (opportunity or actual) cost M_τ per car of that type. This cost may also depend on the request r in order to prioritize terminals. Each region is attributed a track, which is used as reference for distance measures. Then, $c_{i,r} \geq 0$ refers to the distance between region i and the track corresponding to request r. A very rough way to incorporate deadline information is to symbolically define $c_{i,r} = \infty$ for requests r which cannot be fulfilled in time with cars from region i. Other appropriate modifications of the distances enable us to penalize the substitution of car types. Finally, let the non-negative variable $x_{i,r}^\tau$ describe the amount of cars of type τ assigned from region i to request r. A minimum total travel distance allocation of cars to requests is then modeled by a classical transportation problem.

$$\text{minimize} \quad \sum_{i,r,\tau \in \mathcal{T}_r} c_{i,r} \cdot x_{i,r}^\tau + \sum_{r,\tau \in \mathcal{T}_r} M_\tau \cdot x_{S_\tau,r}^\tau \tag{1}$$

$$\text{subject to} \quad \sum_{r:\tau \in \mathcal{T}_r} x_{i,r}^\tau \leq a_i^\tau \qquad \forall\, i, \tau \tag{2}$$

$$\sum_{i,\tau\in\mathcal{T}_r} x^\tau_{i,r} + \sum_{\tau\in\mathcal{T}_r} x^\tau_{S_\tau,r} \geq b_r \quad \forall\, r \tag{3}$$

$$x^\tau_{i,r} \in \mathbb{Z}_+ \forall\, i, r, \tau \in \mathcal{T}_r \tag{4}$$

Constraints (2) prevent exceeding supply of any type in any region, while satisfaction of each demand by an appropriate type, possibly rented, is guaranteed by (3). It is well known that this problem is solvable in integers x in polynomial time e.g., by means of combinatorial algorithms [10].

We remark that one should not try to make suggestions on a very high level of detail. It turns out that such suggestions cannot be implemented in practice. Therefore, our objective function represents only the unavoidable transportation time plus a penalty on renting cars. It may happen in our model that a terminal is supplied with many different types of cars or from many different regions. Forbidding this possibility complicates the problem also from a computational complexity standpoint.

Lemma 1. *It is weakly $\mathcal{NP}$-complete to decide whether a solution to* (1)–(4) *exists with*
1. $x^\tau_{S_\tau,r} = 0$ *for all* r, τ (*no renting*) *and*
2. *for each* $r \in \mathcal{R}$ *there is exactly one pair* (i, τ) *with* $x^\tau_{i,r} > 0$.

Proof: Checking a solution for the required structure is immediate. Thus, the problem is in $\mathcal{NP}$. Consider any collection $\{b_1, \ldots, b_n\}$ of n integers. We construct an instance of (1)–(4) where one region supplies $a^1_1 = 1/2 \sum_i b_i$ cars of type 1 and another region supplies $a^2_2 = 1/2 \sum_i b_i$ cars as well, but of type 2. There are n requests with admissible types $\mathcal{T}_i = \{1, 2\}$ and demand b_i, $i = 1, \ldots, n$, respectively. A solution which satisfies this demand without using cars from some super source induces a partition of $\{b_1, \ldots, b_n\}$, a problem well known to be weakly $\mathcal{NP}$-complete [6]. □

From the modeling point of view it is straight forward to enforce usage of *few*, say W, regions or types as supply for each request by introducing additional binary variables $w^\tau_{i,r}$ and adding to (1)–(4)

$$x^\tau_{i,r} \leq a^\tau_i \cdot w^\tau_{i,r} \quad \forall\, i, r, \tau \in \mathcal{T}_r \tag{5}$$

$$\sum_{i,\tau\in\mathcal{T}_r} w^\tau_{i,r} \leq W \quad \forall\, r \tag{6}$$

$$w^\tau_{i,r} \in \{0, 1\} \quad \forall\, i, r, \tau \in \mathcal{T}_r. \tag{7}$$

On the other hand, theoretically, one easily comes up with instances where this leads to poor solutions. Practically, cars of the same type are not scattered all over the plant, but reasonably concentrated in few regions anyway. However, it may happen that terminals request for several car types at once. Then, all delivered cars should originate from the same, or again, very few regions. Constraints similar to (5)–(7) and individual constraints (3) for each type-request pair model this practically important situation. It follows from Lemma 1 that this problem is $\mathcal{NP}$-hard as well. Also, a single aggregated constraint (6) of

the form $\sum_{i,r,\tau\in\mathcal{T}_r} w_{i,r}^{\tau} \leq W$ can be used to limit the number of simultaneous movements, e.g., in order to reflect a limited number of locomotives. We remark that each loaded car is usually destined for a particular terminal, respectively. When there is choice we pick the loaded cars which are at the plant for the longest in order to return rented cars as quickly as possible. When loaded and empty cars are to be provided simultaneously the loaded cars already fix the region from which the empty cars have to be taken.

3. Shunting minimization

In this section we represent car types by *colors*. Each track is assigned a *head*, which is the principal direction from which the cars are accessed even for two-ended tracks. Thus, a stack is an appropriate concept of describing tracks. Referring to the position of a car on the track we use the notion of *depth*, and say a car is *deeper* in the track when it is further from the head. Accessing a car of depth d implies pulling out all cars on that track up to depth d, a fact we will refer to as *precedence constraint*. We restrict attention to an arbitrary but fixed region. We denote its number of tracks by T; the maximal depth, i.e., the maximal number of cars on a track, is denoted by C.

In each region, when the upper level transportation problem (1)–(4) is solved, the demand $D_i \geq 0$ for each color i is fixed. We face the following *shunting problem*. Each track t is attributed a cost factor $c_t \in \mathbb{Q}_+$, the cost incurred when pulling any *one* car out of the track. An equivalent way of thinking is that the deepest car of each track which is pulled out sums up the cost of *all* cars pulled out of that track, even when not all these are needed to fulfill the demand, see figure 1. The goal is to provide at least D_i cars from each color i at minimal total cost. A practical assumption is that no space limitations apply to intermediate car storage. Further note that neither pushing surplus cars back onto their tracks give rise to any cost, nor exists a prescribed sequence of ordering the cars in the resulting train. Once again, the reasoning is that—unfortunately—a too elaborate scheme cannot be enforced in practice.

Lemma 2. *Deciding whether there exists a feasible solution to the shunting problem at cost at most $K \in \mathbb{N}$ is $\mathcal{NP}$-complete in the strong sense.*

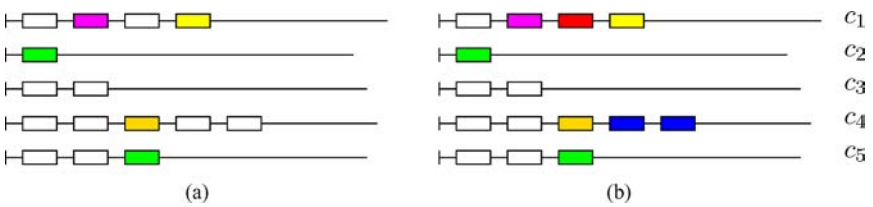

Figure 1. The colored cars in (a) represent the chosen cars of some solution. The cars which incur a cost in this choice are marked in (b). They include those cars inevitably pulled out because of them blocking the way of the demanded cars. The total cost is $3c_1 + 1c_2 + 3c_4 + 1c_5$.

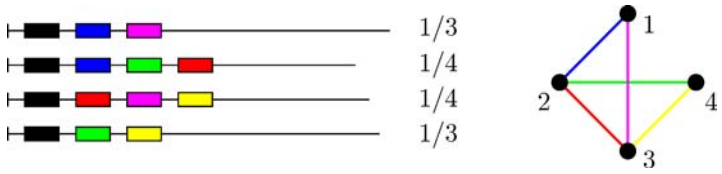

Figure 2. Reducing a vertex cover instance to the shunting problem.

Proof: Determining the cost of a given solution is an obvious polynomial time calculation, and our problem is in $\mathcal{NP}$. Completeness in the strong sense is shown by reduction from vertex cover [6], which is: Given a graph $G = (V, E)$ and a positive integer K, is there a subset $V' \subseteq V$ of at most K vertices such that every edge in E is incident to ("covered by") some vertex in V'? From G we construct an instance of the shunting problem as follows. For each vertex $i \in V$ with degree $\delta(i)$ introduce a track with associated cost $c_i = 1/|\delta(i) + 1|$. Each edge $e \in E$ is interpreted as an individual color. We place a car of color e on track i if and only if edge e is incident to node i. The sequence of placement is of no importance. Bottommost, i.e., deepest on each track we position one car, respectively, in the $|E| + 1^{\text{st}}$ color, say, black. The demand for this instance is given by $D_e = 1, e \in E$, and $D_{|E|+1} = K$. See figure 2 for this clearly polynomial construction.

The purpose of the black cars is to guarantee that we have to clear *all* cars from at least K tracks. Therefore, a solution of cost at most K will use exactly K whole tracks. Now observe that the demand vector ensures that we pick each color at least once. In other words, such a solution induces a vertex cover in G of cardinality K. The reverse direction is now obvious, and the claim follows. □

Shunting minimization remains hard when the maximal depth C of a track is bounded since vertex cover is hard in graphs with bounded degree $\Delta \geq 4$ [6]. The complexity is open when only the number of colors is bounded. However, since the objective function value is exactly preserved in our reduction we obtain a non-approximability result.

Corollary 1. *For our shunting problem, there is no approximation algorithm with guarantee* $7/6 - \varepsilon$ *for* $\varepsilon > 0$, *unless* $\mathcal{P} = \mathcal{NP}$.

Proof: This result is known to hold for vertex cover [7]. Since a factor α approximation algorithm for our shunting problem implies a factor α approximation for vertex cover, the corollary follows. □

3.1. Naïve approaches

A simple greedy strategy is to pick the cheapest car(s) for each color, respectively. It is easy to see that the quality of such a proceeding depends on the chosen picking sequence. Worse, there is no constant factor approximation guarantee for this algorithm. To see this, consider the following instance. We are given n colors, with demand $D_1 = 0$ and $D_i = 1$,

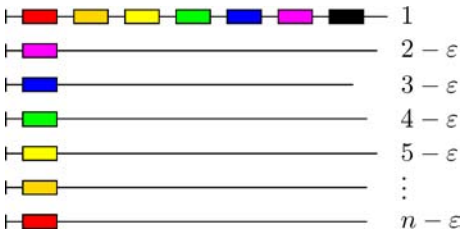

Figure 3. A bad instance for the greedy algorithm.

$i = 2, \ldots, n$. Also, n tracks are available. On the first track, all colors occur in ascending order, i.e., the topmost, easiest accessible color is 1, say black. The cost of this track is $c_1 = 1$. All colors $i = 2, \ldots, n$ except black further occur exactly once again, each on a separate track i at cost $c_i = i - \varepsilon$, with a small $\varepsilon > 0$, see figure 3.

For each color i with positive demand it is now marginally cheaper to pick the respective singleton on track i instead of serving the whole demand at once using the cars on track 1. The latter incurs optimal cost n while the former greedy strategy costs $O(n^2)$. This implies our claim.

Another strategy is to relax the precedence constraints and greedily pick cars ("with a helicopter") from track t at cost c_t. That is, unused cars possibly in the way are pulled out *for free*. Clearly, the accumulated cost H underestimate the optimal cost OPT. Turning this solution into a feasible one by taking into account also the skipped cars results in a cost of at most $C \cdot H \leq C \cdot OPT$. We have obtained a simple factor C approximation algorithm, which unfortunately is the trivial approximation factor.

3.2. *Exact approaches*

First note that one can easily recover a solution to the shunting problem from a vector $(p_1, \ldots, p_T)$ of positions, the tth component $0 \leq p_t \leq C$ of which indicates the depth of the deepest car picked from track t. This suggests implicitly enumerating all possible solutions by way of dynamic programming. The state space is the set of all position vectors; the computational complexity of such an approach is therefore $O(T^C)$. Note that in practical instances obviously T as well as C are bounded.

Actually, we have another practical information on hand, *viz.* the common consecutive occurrence of cars of identical color on a track. We say that cars come in *groups*. It is reasonable to assume that cars picked from a group appear consecutively as well, starting at the head end. In a dynamic program, it is not immediate to us how to make use of this information, and possibly spare states from consideration. Note that C may exceed the maximal number G of groups by an order of magnitude.

Let us now introduce a mixed integer program for the shunting problem. This formulation enables us to better account for cars coming in groups. A pair (t, g) refers to the group of depth $1 \leq g \leq G$ on track $1 \leq t \leq T$. Unless stated otherwise we simplify our notation and assume that indices range in their feasible domains. We denote the size of a group, i.e., the number of cars it is made of, by $Q_{t,g} \geq 1$, and its color by $color(t, g)$. The binary variable

$z_{t,g}$ indicates whether group (t, g) is accessed, and the non-negative variable $y_{t,g}$ expresses how many cars are taken from that group. We would like to

$$\text{minimize} \quad \sum_{t,g} c_t \cdot [(Q_{t,g} - y_{t,g}) \cdot z_{t,g+1} + y_{t,g}] \tag{8}$$

which counts per group (t, g) all used cars $y_{t,g}$ plus those $Q_{t,g} - y_{t,g}$ unused under the condition that the next deepest group $(t, g + 1)$ is accessed as well, i.e., $z_{t,g+1} = 1$. Unfortunately, this intuitive formulation leads us to a nonlinear objective function. Instead, consider the following program.

$$\text{minimize} \quad \sum_{t,g} c_t \cdot Q_{t,g} \cdot z_{t,g} \tag{9}$$

$$\text{subject to} \quad z_{t,g} \le z_{t,g-1} \quad \forall\, t, g > 1 \tag{10}$$

$$y_{t,g} \le Q_{t,g} \cdot z_{t,g} \quad \forall\, t, g \tag{11}$$

$$\sum_{t,g\,:\,\text{color}(t,g)=\tau} y_{t,g} \ge D_\tau \quad \forall \text{ colors } \tau \tag{12}$$

$$y_{t,g} \ge 0 \quad \forall\, t, g \tag{13}$$

$$z_{t,g} \in \{0, 1\} \quad \forall\, t, g \tag{14}$$

The objective function (9) now clearly counts too much, namely all unused cars in the deepest accessed group of each track, respectively. Nevertheless, we claim that we can reconstruct an optimal solution to (8) from an optimal solution to (9). To see this, observe that *some* cars of each respective deepest accessed group *have to* be pulled out in order to fulfill the demand, or else such a group would not have been accessed at all. This implies that the z variables already encode a cheapest allocation of groups. In order to ensure that the y variables assume their smallest feasible values we use a simple greedy strategy which picks the cheapest cars in the chosen groups for each color until the respective demand is satisfied. We argue below that we may remove the y variables from the formulation altogether. Note also that (9) tends to result in *shorter* deepest accessed groups.

Regarding the constraints, (10) encodes the precedence among cars on the same track; (11) guarantees that the supply of an accessed group is not exceeded; the demand of each color is fulfilled due to (12). Constraints (13) and (14) restrict the variables to their domains.

3.3. Putting it all together

When both problem levels are solved still some freedom on how to actually serve requests remains—which is usually considered an advantage by practitioners. In each supplying region trains are made up out of the cars determined in the shunting subproblem and the upper level knowledge about what requests are served by which region. Cars with identical origin and destination regions are transported as one block. In the destination region such trains are split again and moved to the respective terminals without further (major) shunting. All timing ("scheduling") decisions remain up to the planner.

4. An integrated model and extensions

Not least for reasons of benchmarking, it is interesting to simultaneously model the hitherto separated problem levels, i.e., to capture our problem as a whole. Our mixed integer programs combine in a natural way: Only the respective actually shunted cars in (12) are available as supply in (2). Note that we may assume $y_{t,g} = Q_{t,g}$ whenever $z_{t,g} = 1$. Therefore, we substitute $y_{t,g} = Q_{t,g} \cdot z_{t,g}$. Variables z receive a third index $i \in \{1, \ldots, n\}$, indicating the respective region.

$$\text{minimize} \quad \sum_{i,r,\tau \in \mathcal{T}_r} c_{i,r} \cdot x^{\tau}_{i,r} + \sum_{r,\tau \in \mathcal{T}_r} M_{\tau} \cdot x^{\tau}_{S_{\tau},r} + \sum_{i=1}^{n} \sum_{t,g} c_t \cdot Q_{i,t,g} \cdot z_{i,t,g} \tag{15}$$

$$\text{subject to} \quad \sum_{r:\tau \in \mathcal{T}_r} x^{\tau}_{i,r} \leq \sum_{t,g\,:\,color(i,t,g)=\tau} Q_{i,t,g} \cdot z_{i,t,g} \qquad \forall\, i, \tau \tag{16}$$

$$\sum_{i,\tau \in \mathcal{T}_r} x^{\tau}_{i,r} + \sum_{\tau \in \mathcal{T}_r} x^{\tau}_{S_{\tau},r} \geq b_r \quad \forall\, r \tag{17}$$

$$z_{i,t,g} \leq z_{i,t,g-1} \quad \forall\, i, t, g > 1 \tag{18}$$

$$x^{\tau}_{i,r} \geq 0 \quad \forall\, i, r, \tau \in \mathcal{T}_r \tag{19}$$

$$z_{i,t,g} \in \{0, 1\} \quad \forall\, i, t, g \tag{20}$$

Note that any feasible binary solution in terms of the z variables leaves a transportation problem and we still do not need to require integrality of the x variables. The meaning of the constraints is already clear from the above, except the *coupling constraints* (16) which relate the otherwise separated mixed integer programs for each region. In fact, (16) combines (12) and (2). A practical objection against the integrated model may be that the current (and accustomed) structure of solutions is much better reflected by the decomposition approach. This has to be decided by planners. Note that constraints (5)–(7) can be immediately used in this integrated model as well.

One would handle the rejection of expensive requests as follows. First calculate for each request the cheapest way to fulfill it, using our models. This relaxation gives a lower bound. In the case that the demand for some type exceeds the corresponding supply, we reject requests according to non-increasing order of their cost lower bounds. This is particularly easy when the planning is request by request.

Notice that modeling the upper level by a transportation problem only allows for at most *three* consecutive blocks per transportation request. In particular, cars must travel from their origin region immediately to their destination region, without a possible transshipment in another region. Allowing for the latter would result in a multicommodity flow problem. In solving this problem one should exploit that flow is sent only along *very* short paths, i.e., up to three arcs. This problem is interesting from a theoretical point of view, but is not further considered here. An alternative is to use a practical locomotive scheduling approach, where only pre-defined combinations of blocks are allowed [11].

We do not take into account the availability of locomotives since this is a subsequent planning stage. However, we can easily respect the locomotives' capacities cap_i in terms of the number of cars which can be handled per region i (given the planning horizon).

Obviously, adding the following constraints to our models suffice.

$$\sum_{\tau} \sum_{r:\tau \in \mathcal{T}_r} x^{\tau}_{i,r} \leq cap_i \quad \forall\, i \tag{21}$$

Also, by adding constraints similar to (5)–(7) one could limit the total number of regions accessed. This could reflect the number of available locomotives. Both modifications destroy the transportation problem substructure of the model, and integrality of the x variables has to be required explicitly.

Our approach is static, i.e., it does not respect changes over time. If this was demanded in practice we would base the transportation problem on a time-expanded network, using a time discretization of, say, 5 minutes. This is certainly accurate enough and does not dramatically increase the problem size. In the integrated model this issue could become a computational challenge.

5. Computational experience and conclusions

Our practical data come from a small German in-plant railroad which operates at a steel mill. 683 tracks and 168 terminals are organized in 42 regions in which a total of about 1500 cars of 126 possible types are located. One shift of eight hours length comprises 18 transportation requests, a second 49. These instances are named 'de1' and 'de2,' respectively. From these, we deduce three more instances: All cars from both shifts, possibly duplicated, are available in instance 'dens' in order to provoke larger shunting efforts; all requests from both shifts have to be served in instance 'load' in order to allow for a better combination of many requests. In instance 'perm' the cars are randomly permuted on the tracks in order to destroy the manual preordering by the planner.

Using `CPLEX` 8.0 we are able to solve each presented mixed integer program on a standard PC running Linux in two seconds of computation time which is also true for the integrated approach (15)–(20). Tables 1–5 summarize our results. The headings have the following meaning: 'Req' is the number of requests, 'dem' is the total number of requested cars, 'cars' is the total number of cars, 'subst' is the number of substituted cars, 'reg' is the number of regions used to supply, and 'bloc' is the number of positive x variables in the upper level transportation problem (1)–(4), used as an indicator for how many blocks

Table 1. Data specification and characteristics of an optimal solution to the decomposed model.

Instance	Req	Dem	Cars	Subst	Reg	Bloc	Rent	Upper	Lower	LLower
de1	18	113	1575	31	6	28	21	62821	7693	5073
de2	49	324	1458	11	12	56	103	131988	11613	9973
dens	18	113	3033	42	6	27	17	56100	7808	6236
load	68	438	1575	31	11	82	154	179595	13493	12804
perm	18	113	1575	29	11	32	33	56347	12555	9315

Table 2. Results for a simulated manual (FIFO) planning.

Instance	Subst	Reg	Bloc	Rent	Upper	Lower
de1	36	6	28	21	67945	9093
de2	11	12	58	104	138830	15408
dens	30	6	28	17	64852	11836
load	41	11	84	159	204800	19806
perm	33	11	30	36	54381	14804

Table 3. Effect of not using the extended car substitution.

Instance	Subst	Reg	Bloc	Rent	Upper	Lower
de1	5	5	23	44	55691	6769
de2	0	12	54	106	131112	12117
dens	2	6	25	29	58409	7092
load	0	10	72	185	162161	12457
perm	1	10	26	58	42087	10987

Table 4. Effect of limiting the number of supplying regions per request via constraints (5)–(7).

	$W = 1$ in (6)						$W = 2$ in (6)					
Instance	Subst	Reg	Bloc	Rent	Upper	Lower	Subst	Reg	Bloc	Rent	Upper	Lower
de1	28	5	18	24	70078	7017	32	6	24	21	63583	7693
de2	14	11	41	112	115709	10417	10	12	54	106	124080	11613
dens	41	6	18	18	68999	7976	42	6	24	17	57164	8144
load	31	9	57	165	174860	13286	34	11	76	157	172675	13969
perm	27	7	16	41	57233	11680	29	9	26	36	53552	12044

Table 5. Quality of and savings from using the integrated IP (15)–(20).

Instance	de1	de2	dens	load	perm
Total savings	2453	1068	1031	620	3152
Fractional variables %	18.69	5.71	29.72	4.74	0.83
Root node gap %	0.0043	0.0017	0.0217	0.0001	0.00
Root node gap (abs)	90.93	181.50	371.34	28.00	0.00
CPU seconds (decomposed)	1.32	1.47	1.34	1.53	1.56
CPU seconds (integrated)	0.06	0.16	0.36	0.34	0.16

are created. 'Rent' is the number of rented cars, 'upper' is the objective function value of model (1)–(4), and 'lower' is the objective function value of model (9)–(14) summed over all regions. We also compute a lower bound 'llower' on the shunting effort at the lower levels. To this end we set the transportation cost to zero and solve the integrated model (15)–(20).

The cost are scaled such that the order of decreasing importance is rental cost, transportation cost, and shunting cost. Interestingly, when the relative importance of transportation and shunting is reversed or both are equally important, the shunting effort remains almost unchanged. Table 1 lists the basic data specifications for all instances and optimal costs with the decomposed model (1)–(4), (9)–(14). As was to be expected we see a synergy in terms of shunting efforts when more requests are scheduled simultaneously, see the results for instance 'load.' The transportation effort roughly adds up. In Table 2 we emulate a manual, that is, first come first served (FIFO) planning. Each request is scheduled optimally, but one by one separately. We first note that this does not increase the number of blocks, nor the number of rented cars. On average, the transportation cost go up by 10%, and shunting efforts increase by one third. Since we cannot expect to simultaneously plan *all* requests in a shift, and in manual planning also *some* requests are sometimes planned simultaneously, we obtain an interval of possible savings.

The original data explicitly lists allowed car substitutions for every type, also depending on the region. We extend this list in a transitive way, thus building *classes* of similar car types. This is closer to reality, but doing so, we cannot yet compete with the planner's intuition about and experience with substitution, since in the original data *no* cars had to be rented. In Table 2, where we use the original lists, we can see that gathering this information is worthwhile. The fact that there are more rented cars in instance 'perm' than there are in 'de1' is internally due to this lack of information as well.

We test the effect of constraints (5)–(7), i.e., to limit the number W of regions which can be used to fulfill the demand per request, c.f. Table 4. When each request has to be supplied from one region only, either car rental or transportation cost increase significantly. However, already $W = 2$ results in almost the same results as when we do not limit W at all. That is, when we use the basic model the number of supplying regions per request is small anyway.

We finally evaluate the integrated model (15)–(20) in Table 5. The 'total savings' in absolute values have to be interpreted with care since the numbers are relative to our scaling of costs. On the other hand, we save almost the whole difference between the shunting cost 'lower' and their lower bound 'llower' in Table 1. Of course, nothing can be saved at the upper level since the transportation problem is already solved optimally in the decomposed model. Once again, considerable improvement is due to the possibility of a more coordinated way of shunting decisions. As we see from the 'root node gap' the linear programming relaxation gives an excellent lower bound on the optimal integral objective function value, and can be used as an estimate of the quality of some given solution. The integrated model solves even faster than the decomposed one which is largely because of an overhead in the coordination of the two levels. A practical advantage of using the decomposed model is that it can be implemented without the use of a commercial MIP solver.

Concerning the size of our instances we currently do not expect much larger instances even for larger railroads since the amount of work per planner does not significantly increase. Of course, this may change when *all* the planning would be supported by our tool in a coordinated way. Experimental runs of our models with much more requests completed in a few seconds, so we feel prepared.

Our models enable the planner not only to recognize unrealizable requests in time, but also to make optimal use of all available information, in contrast to the current request-by-request planning. Our results suggest that largest savings are to be expected for the shunting cost. One step further would be an optimal shunting also of incoming trains, according to known or expected demand for the respective cars. This could be used to reduce the transportation cost between regions.

Acknowledgments

This research was funded by the German Federal Ministry of Education and Research (BMBF) under grant no. 03-ZIM2BS. Hans-Joachim Lucke and Ulf Hutschenreiter from CSC Ploenzke AG, Dresden, and Wilfried Nittka from Dortmunder Eisenbahn GmbH and CSC Ploenzke AG provided us with valuable insight into the railroad background and with practical data. We thank Cornelia Dangelmayr for stimulating discussions, and an anonymous referee for pointing us to the integrality of x in (19).

References

1. W.P. Allman, "An optimization approach to freight car allocation under time-mileage per diem rental rates," Management Sci., vol. 18, no. 10, pp. B-567–B-574, 1972.
2. C. Barnhart, H. Jin, and P.H. Vance, "Railroad blocking: A network design application," Oper. Res., vol. 48, no. 4, pp. 603–614, 2000.
3. N.J. Bojović, "Application of optimization techniques to the railroad empty car distribution process: A survey," Yugoslav J. Oper. Res., vol. 10, no. 1, pp. 63–74, 2000.
4. A. Charnes and M.H. Miller, "A model for the optimal programming of railway freight train movements," Management Sci., vol. 3, pp. 74–92, 1956.
5. E. Dahlhaus, P. Horak, M. Miller, and J.F. Ryan, "The train marshalling problem," Discrete Appl. Math., vol. 103, nos. 1–3, pp. 41–54, 2000.
6. M.R. Garey and D.S. Johnson, "Computers and intractability—a guide to the theory of NP-completeness," W.H. Freeman and Company: San Francisco, 1979.
7. J. Håstad, "Some optimal inapproximability results," in Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, 1997. ACM Press: El Paso, Texas, pp. 1–10.
8. K. Holmberg, M. Joborn, and J.T. Lundgren, "Computational experiments with an empty freight car distribution model," in Computers in Railways V, J. Allan, C.A. Brebbia, R.J. Hill, G. Sciutto, and S. Sone (Ed.), Computational Mechanics Publications: Southampton, UK, 1996, pp. 511–520.
9. K. Holmberg, M. Joborn, and J.T. Lundgren, "Improved empty freight car distribution," Transportation Sci., vol. 32, no. 2, pp. 163–173, 1998.
10. E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Dower: Mineola, New York, 2001, Unabrigded reprint of the 1976 edition.
11. M.E. Lübbecke and U.T. Zimmermann, "Engine routing and scheduling at industrial in-plant railroads," Transportation Sci., vol. 37, no. 2, pp. 183–197, 2003.
12. H.N. Newton, C. Barnhart, and P.H. Vance, "Constructing railroad blocking plans to minimize handling costs," Transportation Sci., vol. 32, no. 4, pp. 330–345, 1998.

13. Y.-S. Shan, "A dynamic multicommodity network flow model for real time optimal rail freight car management," PhD Thesis, Princeton Unviersity, Princeton, NJ, 1985.
14. H.D. Sherali and A.B. Suharko, "A tactical decision support system for empty railcar management," Transportation Sci., vol. 32, no. 4, pp. 306–329, 1998.
15. T. Winter and U.T. Zimmermann, "Real-time dispatch of trams in storage yards," Ann. Oper. Res., vol. 96, pp. 287–315, 2000.