INFORMS TutORials in Operations Research

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

Network Models in Railroad Planning and Scheduling

Ravindra K. AhujaClaudio B. CunhaGüvenç Şahin

To cite this entry: Ravindra K. AhujaClaudio B. CunhaGüvenç Şahin. Network Models in Railroad Planning and Scheduling. *In* INFORMS TutORials in Operations Research. Published online: 14 Oct 2014; 54-101.
https://doi.org/10.1287/educ.1053.0013

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

 | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0013

Network Models in Railroad Planning and Scheduling

Ravindra K. Ahuja
Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611, ahuja@ufl.edu

Claudio B. Cunha
Department of Transportation Engineering, Escola Politécnica, University of São Paulo, São Paulo, SP, Brazil, cbcunha@usp.br

Güvenç Şahin
Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611, guvencs@ufl.edu

Abstract The past few decades have witnessed numerous applications of operations research in logistics, and these applications have resulted in substantial cost savings. However, the U.S. railroad industry has not benefited from the advances, and most of the planning and scheduling processes do not use modeling and optimization. Indeed, most of the planning and scheduling problems arising in railroads, which involve billions of dollars of resources annually, are currently being solved manually. The main reason for not using OR models and methodologies is the mathematical difficulty of these problems, which prevented the development of decision tools that railroads can use to obtain implementable solutions. However, now this situation is gradually changing. We are developing cutting-edge operations research algorithms, by using state-of-the-art ideas from linear and integer programming, network flows, discrete optimization, heuristics, and very large-scale neighborhood (VLSN) search, that railroads have already started using and from which they have started deriving immense benefits. This chapter gives an overview of the railroad planning and scheduling problems, including the railroad blocking problem, train scheduling problem, yard location problem, train dispatching problem, locomotive scheduling problem, and crew scheduling problem. Some of these problems are very large-scale integer programming problems containing billions or even trillions of integer variables. We will describe algorithms that can solve these problems to near-optimality within one to two hours of computational time. We present computational results of these algorithms on the data provided by several U.S. railroads, demonstrating potential benefits from tens to hundreds of millions annually.

Keywords railroad scheduling; transportation; integer programming; networks; combinatorial optimization; heuristics; modeling

1. Introduction

Transportation is one of the most vital services in the modern economy. In this context, railroads play a significant role by providing efficient and cost-effective freight services for the transportation of products and goods. They also pose some of the most exciting and challenging opportunities for the application of operations research (OR) tools and methodologies.

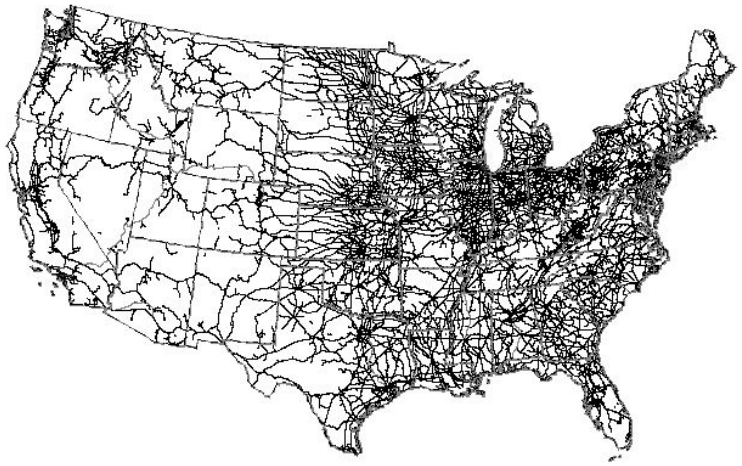
From the supply side, which involves all actions necessary to provide efficient and effective rail freight transportation services, railroad companies face a myriad of decisions and problems at different levels (strategic, tactical, and operational) that are, in general, much more complex than those found in competitive modes, especially trucking and air transportation.

One major difference is that railroads are responsible for building, operating, and maintaining all physical infrastructures related to their respective operations. It encompasses rail tracks on which trains travel and move; stations and yards where trains are formed and, railcars are loaded, unloaded, and switch trains; real-time dispatching, traffic control; and signalization. The number of decisions involved is enormous, usually interrelated, and very complex in nature. These include *strategic, capital investment decisions*, such as expanding the track network, duplicating and increasing capacities of lines; building, improving, and closing yards and stations, acquiring new locomotives, railcars, and other expensive equipments for many uses like loading and unloading; *tactical decisions*, related to train and locomotive schedules and maintenance plans; *operational decisions*, involving train dispatching, yard operations, empty railcar movement, locomotive scheduling, crew management, etc. Other modes of transportation, especially air transportation that has benefited significantly by the use of optimization methods (Yu [50]), do not face the same level of complexity as railroads in terms of its operational environment. Planning and scheduling problems arising in railroads are very large scale and quite complex. For a typical Class I U.S. railroad, its network comprises more than 10,000 miles of tracks, with several thousand weekly trains, and moving about 2,000–3,000 locomotives and no fewer than 80,000 railcars among about 200–300 yards, using some 5,000 train-crew members. Freight railroads operate every day of the year. Figure 1 depicts the U.S. railroad network, comprising over 140,000 miles of rail tracks over which seven major railroads (BNSF, CN, CP, CSX, FXE, KCS, NS, TFM and UP) operate.

From the demand side, each major railroad may receive some 4,000–7,000 new shipments (carloads) from its 5,000 customers every day, to be moved from some 1,000 origins to some other 2,000 destinations. With increasing global competitiveness and customer requirements, railroads must improve the planning and scheduling of their resources to provide fast and reliable services at competitive prices. Immediate actions are necessary for railroads not only to attract new customers, but also to maintain current customers, avoid losing them to truck companies due to poor services, inconsistent, and unreliable transit times, and uncompetitive increasing costs. Evidently, even in this challenging environment railroads also aim to increase their profitability.

This overall scenario is much more complex by far than the one faced by a trucking company or an airline carrier, even a major one. Thus, railroads desperately need computerized planning and scheduling tools so that they can improve their efficiency and profitability. However, the railroad industry has not benefited from the advances in operations research, and still relies mostly on simple simulation-based or manual methods for their planning

FIGURE 1. U.S. railroad network.



and scheduling. The research on railroad optimization problems has grown slowly; most recent contributions deal solely with simplified models or apply them only to unrealistic small instances. The majority of the models developed so far fail to incorporate the complex characteristics of real-life applications (Assad [7], Cordeau et al. [19], Haghani [26], Newman et al. [37]). More recently, the intense competition and the ever-increasing speed of computers have motivated the development and use of optimization models at various levels in railroad organizations. In addition, recently proposed models tend to exhibit an increased level of realism. As a result, there is a growing interest in utilizing optimization techniques for railroad problems. In the last few years, many advances in rail freight and passenger transportation have appeared in operations research literature. Recently, Canadian Pacific Railways was chosen as the winner of the *2003 Franz Edelman Award for Achievement in Operations Research and Management Sciences* for the project "Perfecting the Scheduled Railroad: Model-Driven Operating Plan Development" (Ireland et al. [30]). In this project, OR-based decision-support tools were developed and used to help create a more scheduled railroad, allowing cost savings of about US$ 170 million between 1999 and 2000.

It is important to give the reader a brief idea of what "running a scheduled railroad" means, because it seems to be an appealing approach to the challenging environment that railroads are currently confronted with, particularly increasing customer requirements. The schedule-based approach contrasts the traditional way railroads dispatch trains, known as the tonnage-based approach, which is based on the premise that reducing the number of trains operated, and creating longer trains, will minimize costs. In other words, with the tonnage-based approach trains can be delayed until they have sufficient tonnage to load them to capacity. In this practice, an operating plan may list a train that operates every day, but if there are insufficient railcars, the train can be delayed or even cancelled, with major prejudice to transit times. It causes unreliable services for the customers, may increase operating costs due to reduced utilization of rolling stock assets (locomotives and railcars), and increased idling and deadheading of equipments and crews. In contrast, in the scheduled-based approach trains run on time, as scheduled, even if they travel with light loads.

Though there is still controversy on the subject, this more disciplined approach is gaining favor in North American railroads to craft cost-effective and customer-effective operating plans, as pointed out by Ireland et al. [30]. It should be noted that the impacts in the operation practices are huge, because running lighter and more frequent trains with more rigid transit-time constraints may result in more trains being simultaneously assembled, dispatched, and running; more congested rail tracks and yards, potentially causing delays; more locomotives required to pull more trains; more empty railcars to be moved faster and in smaller batches in order to make them available for loading; and, as a result of all this, a risk of higher overall operating costs. To successfully implement these changes and cope with the increased level of synchronization in this challenging environment, railroads need effective tools to solve their planning and scheduling problems.

This chapter concerns the current state of the art of cutting-edge mathematical models and algorithms for solving some of the most important planning and scheduling problems faced by railroad companies that arise from the above issues in freight transportation. All problems considered here can be seen, and thus modeled, as some type of flow (shipments, trains, locomotives, railcars, crews) over an underlying network. Thus, the focus of this paper is on describing network-based models; the network structure allows us to have a better intuitive understanding of the problem and allows us to develop efficient and effective solution algorithms. A comprehensive review of the main concepts of network flow models and algorithms can be found in Ahuja et al. [4].

We have selected six of the most relevant planning and scheduling problems for railroads that could be effectively solved through network flow OR-based approaches. We give a detailed characterization of each problem, its context and importance to railroads, and the main issues, details, and constraints involved that make real instances extremely difficult

to be modeled and solved from the OR point of view. We also present the mathematical formulations, as well as cutting-edge algorithms to solve these instances arising in practice, and demonstrate significant savings in costs, some in excess of tens of millions of U.S. dollars annually.

The first and foremost planning problem to be solved in railroads is the *blocking problem*. This problem arises in the context that a railroad carries *millions* of shipments from their origins to their respective destinations. A typical shipment is composed of a set of individual *cars* that all share a common origin and destination. To reduce the handling of individual shipments as they travel, a set of shipments is *classified* (or grouped) together to create a *block*. The railroad blocking problem consists of identifying this blocking plan for all shipments in the railroad network so that the total transportation and handling cost is minimized.

The *yard location problem* is closely related to the railroad blocking problem. In this problem, we aim to find the best network configuration in terms of the number and location of yards where cars can be reclassified into new blocks and switch trains. Its importance is derived from the fact that yard locations have a major impact on blocking plans. However, in general, yard locations have been determined historically and have not changed much despite the significant cost implications, mostly fixed costs associated with equipment, infrastructure, and labor.

Given a blocking plan, developing a *train schedule* is perhaps the next-most important operational planning task faced by a railroad. The train scheduling problem is to determine train routes, their frequencies, and the days of operation of each train, aiming to minimize the cost of carrying blocks of cars from their origins to their destinations.

The *locomotive scheduling problem* consists of efficiently assigning different types of locomotives to the scheduled trains so they receive the desired pulling power while satisfying a variety of constraints, including fleet-size constraints on different locomotive types, and fueling and maintenance constraints.

The *train dispatching problem* aims to determine detailed train movements and timetables over a rail network under various operational constraints and restrictions in order to minimize both train deviations from the planned schedule and total delay as well. It is mostly motivated by the fact that trains traveling on single-line tracks can only overtake and cross each other at specific locations (sidings or meet-points), which are conveniently located at regular intervals along the line. Delays or deviations occur when trains traveling either in opposite directions or in the same direction meet, thus requiring one of the trains to be pulled over for the other to cross or overtake it.

The *crew scheduling problem* entails assigning crews to trains for each crew district, while complying with union rules, so that the crew costs are minimal and train delays due to crew unavailability are minimized.

In the following sections, each of the above problems is described in detail.

2. Railroad Blocking Problem

The *railroad blocking* problem is essentially a consolidation problem—how to consolidate a large number of shipments into *blocks* of shipments so as to reduce their individual handling as they travel from their origins to their destinations. Variants of this problem arise in several transportation sectors, including postal and package delivery companies, trucking companies, and the airline industry.

We will illustrate this through a problem faced by a large postal carrier such as the U.S. Postal Services (USPS). Consider, for example, the post office at a small city, which may receive several thousand envelopes every day to be mailed to thousands of other cities across the United States. It should be noted that handling each envelope individually—that is, sending it directly to its destination—would be highly inefficient because it would require

creating a separate bag for each of thousands of destinations and handling that many bags (some of which may be very small) as they travel to their destinations. Thus, the post office sorts (or consolidates) these individual envelopes into a small number of postal bags destined for different regions in the United States. For example, the post office may make a bag for Los Angeles (L.A.), and all envelopes headed for California can be placed into that bag. L.A. may receive several hundred bags from different cities in the United States that will be opened and resorted, and new bags will be created. At L.A., the post office may create a mailbag for Palo Alto; this bag will contain mail arriving at L.A. from several origins across the nation, all destined for Palo Alto.

Therefore, the purpose of the sorting process is to reduce the handling of the individual shipments to be moved. This sorting (or classification) process is a standard process in many situations where a large number of commodities travel over a network between several different origin-destination (OD) pairs of nodes. It arises in LTL (less than truckload) trucking industry where a large number of small shipments are consolidated into truckloads. An airline carrier's hub-and-spoke configuration also consolidates passenger traffic between thousands of origin-destination pairs into flight legs flown by different planes. It also arises in telecommunications where voice or data messages are consolidated and routed over the designed telecommunications network.

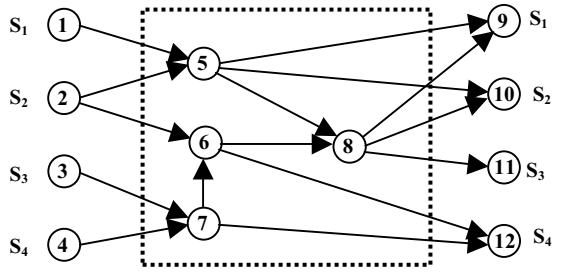
In railroads, this consolidation problem arises in the contexts of general carload (also known as general merchandise) and intermodal services. Carload services focus on the movement of small lots of railcars belonging to many different customers from a diverse set of origins to a diverse set of destinations. Intermodal transportation involves the movement of freight containers and over-the-road truck trailers on specially designed railcars between many different OD pairs. Both types of services are driven by the customer's need for shorter transit times and lower tolerances for service variances and delays. In these services, however, the amount of freight to be shipped from an origin station to a destination station does not usually allow the dispatching of a unit train, i.e., an entire train that is moved from the loading station to the unloading station without any intermediate handling, and usually containing a single product or a shipment for a single customer.

The number of shipments to be moved by railroads may be very high (about 50,000 in a month for a typical U.S. railroad). Given that it is physically and also economically impossible for a railroad to transport and handle each shipment individually, railroads have to consolidate shipments and respective railcars, route them across the network in blocks, through the rail yards and on the trains, in order to efficiently and effectively move them from their origins to their destinations. Due to this consolidation, some shipments may have to travel a longer distance with respect to their shortest-path direct routes from their respective origins to their destinations, as well as go through some intermediate handlings and switch trains (called *block swap*).

A railroad block is composed of several incoming and originating shipments that are grouped together. It is defined by an OD pair that may be different from the OD pairs of individual shipments in the block. Once a shipment is placed in a block, it will not be classified again until it reaches the destination of that block. Therefore, the blocks to be made are design arcs in a network representation. The railroad blocking problem is to construct the blocking network so that when all rail shipments are optimally routed over the blocking network, the total transportation cost, which is the weighted sum of the costs of distance traveled by cars and intermediate handlings, is minimum.

Figure 2 shows a sample blocking network, where three types of nodes can be identified: *origins* (nodes 1–4), where shipments S_1, S_2, S_3, and S_4 originate; *yards* (nodes 5–8), where shipments can be reblocked or reclassified; and *destinations* (nodes 9–12), where shipments S_1, S_2, S_3, and S_4 terminate. Each arc in the network represents a *block* with the origin at the tail of the arc and destination at the head of the arc. For example, arc (1, 5) signifies that a block will be made from station 1 to station 5. Shipments travel from their origins

FIGURE 2. An example of a blocking network.



to destinations over the arcs built in this blocking network. For example, a shipment S_1 that originates at node 1 and is destined for node 9 can be routed on two blocking paths: 1-5-9 and 1-5-8-9. The first blocking path flows on arcs (1, 5) and (5, 9) and involves one intermediate handling at node 5; the total cost is the sum of the flow costs on the two arcs with the handling cost at node 5. The second blocking path flows on arcs (1, 5), (5, 8), and (8, 9) and involves two intermediate handlings: at nodes 5 and 8. Another alternative, for example, would be to create a block arc (10, 9) linking node 10 to node 9. Similarly, a block arc (1, 8) would link node 1 directly to node 8 for all those shipments that flow on arc (1, 8), without intermediate handling at the yard located at node 5, even if this node is in the physical path between nodes 1 and 8. It should be noted that, in practice, yards can be origins as well as destinations, and nodes can send and receive shipments as well.

We shall now give the reader a rough estimate of the annual blocking cost a major U.S. railroad incurs and the potential savings that can be achieved. For a typical railroad, the flow cost per car per mile is \$0.50, and the average cost of reclassification is \$40 per car. A typical shipment may travel an average of 500 miles from its origin to its destination and may be classified 2.5 times. Assuming that the railroad ships 50,000 shipments per month with an average of 10 cars per shipment, we obtain 6 million cars per year traveling a total of 3 billion car-miles and 9 million reclassifications per year. This gives us a total annual cost of \$1.5 billion for car-miles traveled and \$1 billion in classifications. If these costs can be reduced by 2% to 5% by the use of an optimization-based algorithm, the annual savings will be of tens of millions of dollars for a single U.S. railroad and several hundred millions of dollars for all U.S. railroads.

2.1. Mathematical Formulation

Mathematically, the railroad blocking problem can be modeled as a *network design multi-commodity flow problem*. In this problem, we have to select blocking arcs to build, and also decide how each shipment k of a set K, consisting of v_k railcars with the same origin $o(k)$ and the same destination $d(k)$, is grouped with other shipments in blocks and moved over these blocking arcs by different trains through some classification yards where blocking operations (i.e., the grouping of incoming cars for connection with outgoing trains) are performed.

Let $G = (N, A)$ be the blocking network, where N is the set of all nodes denoting the stations where shipments originate, terminate, or switch trains and A is the set of all potential blocking arcs in $N \times N$, i.e., $(i, j) \in A$ if a block can be built from node i to node j. The cost of classifying a car at node i is h_i and the unit flow cost per car of shipping a block through arc (i, j) is m_{ij}.

The blocking problem requires that all v_k cars in each shipment $k \in K$ must be sent along a unique path in the blocking network. An efficient blocking solution needs to restrict the number of blocks made at a node $i \in N$ (given by b_i), as well as the number of cars it can handle (denoted by d_i). The maximum number of cars that can flow on block $(i, j) \in A$ is given by u_{ij}. These approximate estimates of yard capacities can be determined by its geometric configuration. There is a body of literature devoted exclusively to modeling yard

operations at a much more microscopic scheduling level, not taken into consideration in blocking modeling. We refer the interested reader to the survey work of Cordeau et al. [19].

The blocking problem has two sets of decisions: *design variables* $y_{ij} \in \{0,1\}$, concerning which links or arcs $(i,j) \in A$ to build; and *flow variables* x_{ij}^k, concerning how commodities should flow over the arcs in the network. In other words, $x_{ij}^k = v_k$ if shipment k flows on the arc $(i,j) \in A$; 0 otherwise. Flow variables depend upon the design variables, as flow can take place only on the arcs that are built. The sets of arcs in A entering and emanating from node i are given by $I(i)$ and $O(i)$, respectively. The blocking problem can be formulated as the following mathematical programming problem.

$$\text{Minimize} \quad \sum_{k \in K} \sum_{(i,j) \in A} m_{ij} x_{ij}^k + \sum_{i \in N} \sum_{k \in K} \sum_{(i,j) \in O(i)} h_i x_{ij}^k \tag{1}$$

$$\text{subject to} \quad \sum_{(i,j) \in O(i)} x_{ij}^k - \sum_{(i,j) \in I(i)} x_{ij}^k = \begin{cases} v_k & \text{if } i = o(k) \\ 0 & \text{if } i \neq o(k) \text{ or } d(k), \\ -v_k & \text{if } i = d(k) \end{cases} \quad \forall k \in K \tag{2}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \quad \forall (i,j) \in A \tag{3}$$

$$\sum_{(i,j) \in O(i)} y_{ij} \leq b_i, \quad \forall i \in N \tag{4}$$

$$\sum_{k \in K} \sum_{(i,j) \in I(i)} y_{ij} \leq d_i, \quad \forall i \in N \tag{5}$$

$$y_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{6}$$

$$x_{ij}^k \in \{0, v_k\}, \quad \forall (i,j) \in A, \forall k \in K \tag{7}$$

In the above formulation, the objective function (1) seeks to minimize the weighted sum of flow costs on the arcs and classification costs at the nodes. The constraint (2), in conjunction with the constraint (7), ensure that each shipment flows on a single and unique path in the blocking network, and the constraint (3) ensures that shipments can flow on an arc only if it is built and the total flow on this arc is less than or equal to its flow capacity. The constraint (4) restricts the number of blocks created at a node and the constraint (5) restricts the number of car handlings at any node. Constraint (6) ensures that the arc design variables are binary variables.

The railroad blocking problem is a very large-scale network optimization problem. To illustrate this, let us consider a large U.S. railroad. It may have over 1,000 origins, over 300 yards, and over 2,000 destinations, and may send over 50,000 shipments in a month (a shipment may contain several cars usually ranging from 1 car to 50 cars). We can create a block from any origin to any yard, from any yard to any other yard, and from any yard to any destination. This gives a total of $1{,}000 \times 300$ (origins to yards) $+ 300 \times 300$ (yards to yards) $+ 300 \times 2{,}000$ (yards to destinations) ≈ 1 million potential blocks that can be build. As there are over 50,000 shipments and about 1 million potential blocks, the number of decision variables for routing the shipments are in billions. Hence, this blocking problem contains about a million network design variables (either build or not build a block) and billions of flow variables (corresponding to the routing decisions of shipments over the blocking network).

Several attempts have been made in the past to model and develop an optimization-based algorithm to solve this problem (Barnhart et al. [8], Bodin et al. [9], Newton [38], and Newton et al. [39]). The largest problems solved so far are much below the size of real-world blocking problems found in practice. In addition, the optimization-based solution approaches described in the literature either do not produce high-quality solutions, or do not satisfy all the practical considerations necessary for implementability. To summarize, in spite of all achievements and advances in terms of optimization-based approaches, the

largest problems solved so far are much below the size of real-world blocking problems found in practice. However, a neighborhood search algorithm that has recently been proposed to solve the blocking problem appears to be very promising. We describe this algorithm next.

2.2. A Neighborhood Search Algorithm

A recent paper by Ahuja et al. [2] proposes very large-scale neighborhood (VLSN) search algorithms to solve real-life instances of the blocking problem. The VLSN search algorithms are neighborhood search algorithms where the size of the neighborhood is very large; in fact, it is so large that enumerating and evaluating all neighbors is prohibitively expensive. Using concepts from network flow theory (Ahuja et al. [4]), a VLSN search algorithm implicitly enumerates a vast neighborhood to identify an improved neighbor. We refer the reader to the paper by Ahuja et al. [5] for a detailed description of VLSN search algorithms.

In order to solve the railroad blocking problem, we make a simplifying assumption that makes it much easier to solve. We assume that the blocking problem needs to honor only the blocking capacities of nodes (b_i); the other capacity constraints—that is, car handling (d_i) and flows on arcs (u_{ij})—can be ignored. These assumptions are based on the fact that in real-life blocking problems, even if these constraints are ignored, most nodes and arcs automatically satisfy them. Thus, in the absence of these capacity constraints, we need to honor only the constraint on the number of blocking arcs built at any node (b_i). Further, once we have decided what blocks will be built, all shipments can travel along their shortest paths from their origins to their respective destinations in the blocking network without violating any constraint. Hence, decision variables in the blocking problem reduce to what blocks should be built (i.e., which y_{ij}-variables equal 1), and thus flow variables x_{ij}^k are automatically determined, because all shipments are always routed along their shortest paths in the blocking network built. The dropped constraints, i.e., flow capacity of blocking arcs and car-handling capacities at nodes, are handled by adding some Lagrangian relaxation-based penalties on their violation.

Figure 3 describes the VLSN search algorithm to solve the railroad blocking problem. It starts with a feasible solution of the blocking problem and iteratively improves the current blocking solution by replacing it by its neighbor until it cannot be improved. The neighborhood of a blocking solution is defined as consisting of all the blocking solutions that we can obtain by changing the blocks at one node of the blocking network. This algorithm has two important subroutines: constructing the initial feasible solution, and reoptimizing the blocking arcs emanating from a node. Each of them is described in detail below.

2.2.1. Obtaining a Feasible Initial Solution. We use a fairly simple construction heuristic to create the initial solution of the blocking network. This solution must ensure that the origin node for each shipment is connected to its destination node through a blocking path. To ensure this, a directed cycle that passes through all the yards exactly once and returns to the first yard is constructed (that is, a Hamiltonian cycle). Then, a blocking arc from each origin node (where some shipment starts) to the nearest yard is created.

FIGURE 3. The VLSN search algorithm for the railroad blocking problem.

```
algorithm VLSN-Blocking;
begin
    construct an initial blocking network and send all shipments along shortest paths in the
    blocking network;
    while the current solution is not locally optimal do
      for each node i ∈ N do {one pass}
          reoptimize the blocks emanating from node i;
          send all shipments along shortest paths in the updated blocking network;
      end; {one pass}
end;
```

Next, each destination node (where some shipment terminates) is connected to the nearest yard that still has some spare blocking capacity. After the blocking network is constructed, shipments are routed along their shortest paths over this network. In their computational experiments, Ahuja et al. [2] found that the proposed algorithm is fairly insensitive to the starting solution, and the effort to obtain a better initial solution does not have much impact on the final solution.

2.2.2. Improvement Procedure. An initial feasible solution is improved by exploring its neighborhood. The neighborhood of a blocking solution is defined as all the solutions that can be obtained by changing the blocks made at only one node and then rerouting all shipments along their updated shortest paths, which corresponds to the loop {*one pass*} of the algorithm in Figure 3. This neighborhood is very large, as there are n ways to select the node for which we change the blocks emanating from it. Secondly, the number of ways we can select k blocks out of a node with p arcs emanating from it is ${}^{p}C_{k}$, which grows exponentially with p and k. Clearly, enumerating all neighbors of a blocking solution and evaluating them explicitly is out of question.

Thus, we observe that the problem of identifying the best neighbor in the neighborhood (or a better neighbor than the current solution) is a difficult task. The formulation of this neighborhood problem as an integer programming problem could not be solved to optimality or near-optimality using state-of-the-art commercial integer programming software in several hours of computing time. This problem is believed to be NP-complete and, therefore, a heuristic solution is required. The paper Ahuja et al. [2] develops an algorithm that finds an improving neighbor heuristically. One pass of the improvement procedure for a selected node is composed of the following steps:

Step 1. Delete all blocking arcs emanating from the node.

Step 2. Reroute the traffic passing through the node along new shortest paths.

Step 3. Build new blocking arcs one at a time, causing maximum savings in the total cost.

Step 4. Reroute the shipments originating at (or passing through) this node over the shortest paths with the new blocking arcs.

As a result of Steps 1 and 2 of this algorithm being executed for a node i, all blocks made at this node are reoptimized, assuming that the blocks at other nodes do not change. The following *maximum savings rule* is considered in Steps 2 and 3: For every potential blocking arc that can be built at node i, it computes the savings in the total cost if that blocking arc is built and all shipments are rerouted to take advantage of this new blocking arc, and selects the blocking arc with the maximum savings. Step 4 reroutes all the shipments for which costs would decrease by using the newly built blocking arc.

2.2.3. Handling Additional Practical Constraints. The proposed VLSN search algorithm can also handle a variety of operational constraints and features that may arise due to historical restrictions, customer commitments, or trade-union-related reasons faced by railroads that cannot be easily incorporated into optimization-based formulations. These constraints and features include: (i) *blocks that must be made*, due to historical reasons and customer commitments; (ii) *blocks that must not be made* because they may be considered undesirable from crew, locomotive, or geographical-related issues; (iii) *some shipments must be sent on pre-specified blocks*; (iv) *not building small blocks*, because railroads do not like building small blocks when they travel large distances, as each block requires a certain amount of cost to build, and to subsequently handle it at yards, and small volume blocks do not justify this cost; (v) *give preference to some blocks* in order to avoid changing the blocking plan dramatically every month in response to variations in demand; (vi) *generate a blocking plan consistent with the train schedule*, that is, requiring that a block may not swap more than a certain number of trains, as each train swap requires some handling cost and delays the shipments in the block; (vii) *allow making incremental changes in a given*

blocking solution, because railroads may not be willing to make dramatic changes due to their associated risks in contrast with minor changes that can be easily implemented; (viii) *optimize a part of the blocking network*, which may be some selected nodes or a part of the railroad corridor.

Further details of the VLSN search algorithm can be found in Ahuja et al. [2]. We now describe the results of computational experiments for some real-world blocking problems.

2.3. Practical Experiments

The proposed VLSN search algorithm has been tested on real-world data provided by three major U.S. railroads: CSX Transportation, BNSF Railway, and NS Corporation. The results focus on comparing the blocking solutions of the algorithms with the blocking solutions currently used by the railroads. Two statistics are used to compare the solutions: average miles traveled by a car (which measures the cost of flow) and the number of reclassifications of cars (which measures the reclassification costs).

The first set of computational results compares VLSN search solutions with the corresponding railroads current solutions when it is allowed to make any block in the blocking network; it demonstrates, to some extent, the maximum savings possible due to changes in the blocking network. Table 1 gives these results for three railroads. The computational times given are taken on a 2.4 GHz Pentium IV computer with 2 GB RAM. We observe that the intermediate handlings can be dramatically reduced by using an optimized blocking plan.

We next report the computational results of the VLSN search algorithm when the blocking solution is restricted by the current train schedule, that is, requiring that a block may not swap more than a certain number of trains. In these tests, we prohibit those blocks that require more than two block swaps. Table 2 gives these results for two railroads.

The paper reports other results where incremental changes are made to a given blocking solution. The results are quite expressive. For instance, for CSX data, a change of only 5% in the current blocks results in 3.2% savings in car miles and 15.2% in reclassifications when solutions are unrestricted by the train schedule; and 2.9% and 12.6%, respectively, when solutions are restricted by the train schedule.

2.4. Concluding Remarks

The blocking problem is one of the most important railroad blocking problems, and its difficulty precluded the development of any optimal or near-optimal algorithm. As a result, almost all the railroads currently solve this problem manually. The VLSN search algorithm described here has a real potential to solve the blocking problem effectively and to be used in practice by railroads. It is also very flexible, and can easily incorporate a variety of practical considerations. The computational results of the algorithm are very impressive. For the data provided by the three major U.S. railroads, significant improvements have been obtained in short computing time. Another surprising outcome of the computational testing is that even a minor change in the blocking solution can lead to fairly dramatic improvement in reclassifications.

In the next section, we describe the yard location model, closely related to the railroad blocking problem because location and number of yards significantly affect how the blocking network is built, how and where blocks are made, and consequently, the detour of cars as

Table 1. Comparison of solutions unrestricted by the train schedule.

	CSX	BNSF	NS
Savings in average car miles	4.8%	0.1%	2.0%
Savings in reclassifications	31.6%	32.7%	41.9%
Computational time	2 hours	5 minutes	1 hour

Table 2. Comparison of solutions restricted by the train schedule.

	CSX	BNSF
Savings in average car miles	4.4%	5.2%
Savings in reclassifications	23.2%	19.8%
Savings in block swaps	3.6%	4.7%
Computational time	2 hours	5 minutes

they travel from their origins to their destinations over the railroad network. In this problem, we aim to find the best network configuration in terms of the number and location of these yards.

3. Yard Location Problem

Yards play a vital role and can be considered the most crucial nodes of the blocking network because yards are where cars from inbound trains are reclassified and assembled into outbound blocks. Hence, these yards are usually referred as the nerve centers of the railroad system. From the definition of the blocking problem, it is immediately clear that the spatial configuration of yards is a very critical factor in identifying the resulting blocking network. Location and number of yards significantly affects the routes of cars and their detours as they travel in blocks from their origins to their destinations over the railroad network.

Yards are of three types: local, system, and regional. *Local yards* are small-scale yards where shipments from nonyard stations are assembled into blocks. *System* and *regional yards* are classified as hub yards (or hubs) that are occupied with large handling capacities. The yard location problem deals with the location of such yards. A major U.S. railroad has about 20–40 hub yards. Most of them were established several decades ago, considering the traffic pattern at that time. Due to changes in the traffic pattern through years, traffic growth, and railroad mergers, many yard locations are far from optimal. This implies that even the optimized blocking network with the current yard configuration would not give the best possible and least costly alternative to handle and route the shipments, unless the number of yards and their locations are reviewed. Costs involved in building a new yard, expanding the capacity of an existing yard, and operating the yards constitute a significant portion of capital investment. Hence, a railroad cannot afford to operate without analyzing the locations of its yards. This problem turns out to be the topmost strategic-level problem for a railroad in the process of finding the optimal way to ship goods from their origins to their destinations.

North American railroads have never paid attention to this strategic problem in the past. Even after the period of mergers, railroads have chosen to keep the yard locations as they are. As railroads have never studied the benefits of rearranging their yard configuration, the research on this particular issue has not attracted attention in the operations research literature. In this section, we discuss the first novel attempt to model and solve this problem.

Railroads potentially have much to gain if they can arrive at answers to the following questions:

(i) If some major yards need to be shut down, then what are the best yards to close with minimal impact on the transportation cost?

(ii) What are the trade-offs between the number of major yards and the transportation costs?

(iii) If some new yards are to be opened, then how many should be opened, and where?

(iv) If capacities of some yards can be increased, where should this capacity expansion should be done?

(v) What is the best network configuration? What are the impacts of operating a more centralized network in terms of fewer hub yards with increased capacities?

As railroads pay more attention to this strategic problem, they understand that some of the current major yards can be closed with little or no impact on the transportation cost, or some of the major hub yards can be relocated to significantly reduce the transportation costs. It is clear that any change in the yard locations would change the shipment plan of a railroad, because losing a current yard would invalidate the current blocking plan, or establishing a new yard location would make an improved blocking plan possible.

3.1. Mathematical Formulation

The yard locations are a major input to the railroad blocking problem, and the immediate effect of changing the yard locations will be changes in the blocking network. From this observation, it is obvious that yard location decisions should be considered along with their effects on the blocking network. Therefore, any mathematical formulation of the problem should include the blocking problem. The formulation approach we present here (Ahuja et al. [3]) is built on the same mathematical formulation of the blocking problem (Ahuja et al. [2]) presented in §2, with the addition of a high-level binary decision variable that identifies the selected yard locations in a given set of candidate yard locations. In general, similar to a simple facility location problem, the number of selected yard locations can be restricted either by a budget constraint related to the total cost of establishing and operating the yards, or by including these costs in the objective function. Here, we assume that the desired number of yards is known by the railroad and restricted to be less than that particular number (p). Without loss of generality, we assume a subset $\bar{N}$ of all yards N that contains all candidate yard locations, and let z_i equal 1 if we select a yard at node $i \in \bar{N}$ and 0 otherwise. Let $\bar{b}_i$ denote the blocking capacity of a candidate yard $i \in \bar{N}$ (that is, the maximum number of blocks that can be made at yard i) and $\bar{d}_i$ be the car-handling capacity of yard i—that is, the maximum number of cars that can pass through yard i. This yard location problem has the following formulation:

$$\text{Minimize} \quad \sum_{k \in K} \sum_{(i,j) \in A} m_{ij} x_{ij}^k + \sum_{i \in N} \sum_{k \in K} \sum_{(i,j) \in O(i)} h_i x_{ij}^k \tag{8}$$

$$\text{subject to} \quad \sum_{(i,j) \in O(i)} x_{ij}^k - \sum_{(i,j) \in I(i)} x_{ij}^k = \begin{cases} v_k & \text{if } i = o(k) \\ 0 & \text{if } i \neq o(k) \text{ or } d(k), \quad \forall k \in K \\ -v_k & \text{if } i = d(k) \end{cases} \tag{9}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij} \quad \forall (i,j) \in A \tag{10}$$

$$\sum_{(i,j) \in O(i)} y_{ij} \leq b_i \quad \forall i \in N \backslash \bar{N} \tag{11}$$

$$\sum_{k \in K} \sum_{(i,j) \in I(i)} y_{ij} \leq d_i \quad \forall i \in N \backslash \bar{N} \tag{12}$$

$$\sum_{(i,j) \in O(i)} y_{ij} \leq b_i + \bar{b}_i z_i \quad \forall i \in \bar{N} \tag{13}$$

$$\sum_{k \in K} \sum_{(i,j) \in I(i)} x_{ij}^k \leq d_i + \bar{d}_i z_i \quad \forall i \in \bar{N} \tag{14}$$

$$\sum_{i \in \bar{N}} z_i \leq p \tag{15}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{16}$$

$$x_{ij}^k \in \{0, v_k\} \quad \forall (i,j) \in A, \ \forall k \in K \tag{17}$$

$$z_i \in \{0,1\} \quad \forall i \in \bar{N}. \tag{18}$$

Among the existing constraints in the blocking formulation, the block-making (11) and car-handling (12) capacity constraints are now only defined for those locations that are not candidate yard locations. For candidate yard locations, constraints (13) and (14) ensure that the intermediate handlings can be performed only at those locations where yards have been located, and at other locations only originating or terminating shipments can be handled. Constraint (15) restricts the number of yards, and constraint (17) captures the integrality constraint for the new location variable. All other constraints are the same as in the blocking-problem formulation.

As seen in the previous section, the blocking formulation cannot be solved to optimality for a real-life problem because it contains too many variables and constraints. It is clear that the yard location formulation with an additional binary decision will be even more difficult to solve using commercial software. Hence, we need to develop heuristic algorithms to solve this problem. We describe one such algorithm below.

3.2. Solution Method

The solution method for the yard location problem (Ahuja et al. [3]) is based on the solution algorithm of the blocking problem (Ahuja et al. [2]) presented in §2. The analysis requires using the blocking algorithm as a subroutine. This subroutine may be executed several thousand times to perform a complete analysis over the current set of yard locations of a railroad. Therefore, we first develop a simplified and specialized version of the blocking problem that can be solved very efficiently. Using this specific implementation for the blocking problem, we develop algorithms to make decisions about closing and opening yards.

In the simplified version of the blocking problem, the assumptions result in a significant reduction in the problem size without sacrificing the accuracy of the solution with respect to the effects of yard configuration. The algorithm to be used as a subroutine in yard location analysis solves this simplified version of the railroad blocking problem, but in essence would find the major blocking network structure given a set of yards, yard capacities, and shipment data.

For this simplified version of the blocking problem we make the following assumptions:

(i) If a shipment has its origin/destination at a customer location, its origin/destination is mapped to the nearest yard (system, regional, or local). Observe that after this mapping, all traffic originates and ends at yards (system, regional and local).

(ii) We allow intermediate block switching only at hub yards. Local yards can build, send, and receive traffic, but do not allow intermediate switching.

(iii) We build mandatory blocks between every pair of hub yards. In addition, each local yard sends and receives a block from the nearest hub yard. A local yard can also build a block to another hub yard, provided there is sufficient car volume or that building such a block will result in substantial savings in car miles.

It should be noted that a key feature of the solution approach is to allow railroads to assess possible benefits of decreasing the number of yards in their system. Decreasing the number of yards in the rail network reduces the costs associated with operating yards (which should be considered as fixed-charge costs per year or per lifetime of the yard), and simultaneously increases the costs associated with routing the shipments over the rail network (which should be considered as variable costs per shipment) as the shipments might have to take longer routes when there are fewer yards for blocks to be made or switched. Another desirable feature would be to evaluate the trade-off between the savings associated with operating fewer yards and the increase in transportation costs due to running trains on longer routes. Railroads do not necessarily want to solve the problem to optimal yard locations, but rather want to benefit from this approach as a "what-if" analysis tool. As the problem on hand is a highly strategic one, they want to analyze the effects of changes in the most important inputs to the problem, such as number of yards, yard operating and establishment costs, and yard capacities. Thus, the overall analysis will be capable of showing the results with

different number of yards and with different cost structure settings, and hence help the decision maker assess the best setting. In the following section, we discuss the results of:

(i) a *drop algorithm* that would consider eliminating each yard location one by one and drop the yard with minimal impact on the cost;

(ii) an *add-drop algorithm* that would first eliminate a set of yard locations, then consider adding several yard locations one by one and add the yard with the maximum savings in costs; and

(iii) an *exchange algorithm* that would consider swapping an existing yard location with a potential yard location and perform the swap with the maximum benefit.

The *drop algorithm* starts with solving the blocking problem and sorting the hub yards in the ascending order of their switching volumes. It then picks the first k hub yards in this order, deletes each of these hub yards one by one, and considers the impact on the solution statistics. The hub yard whose deletion results in the best overall statistics is dropped. We again re-sort the hub yards in the ascending order of their updated switching volumes, consider the new set of the first k yards for deletion, and drop the yard with least impact on the overall statistics.

In the *add-drop algorithm*, we consider eliminating several yards one after another and adding some of them back to eventually find a solution with fewer yards than the initial solution. We drop p hub yards one by one and add back q hub yards ($p > q$) one by one. We always drop the hub yard that worsens the overall statistics by the minimum amount and add the hub yard that improves the overall statistics by the maximum amount.

The *pairwise exchange* algorithm attempts to improve a solution by exchanging a current hub yard with a nonhub yard. This algorithm can be used to improve a solution with a particular number of yards obtained by either the drop algorithm or the add-drop algorithm.

3.3. Computational Experiments

The computational experiments are from a data set provided by a major U.S. railroad with 27 current yard locations. The data set comprises 77 days of traffic and contains about 1.2 million cars. The railroad wants to make an analysis in order to close some of its current yards, and hence finds the optimal number of yards to keep and their respective locations. In order to help the railroad management to find the set of yards to keep in their system, previously mentioned algorithms are implemented as follows:

(i) In each iteration of the drop algorithm we consider the deletion of a subset of current yards of size k.

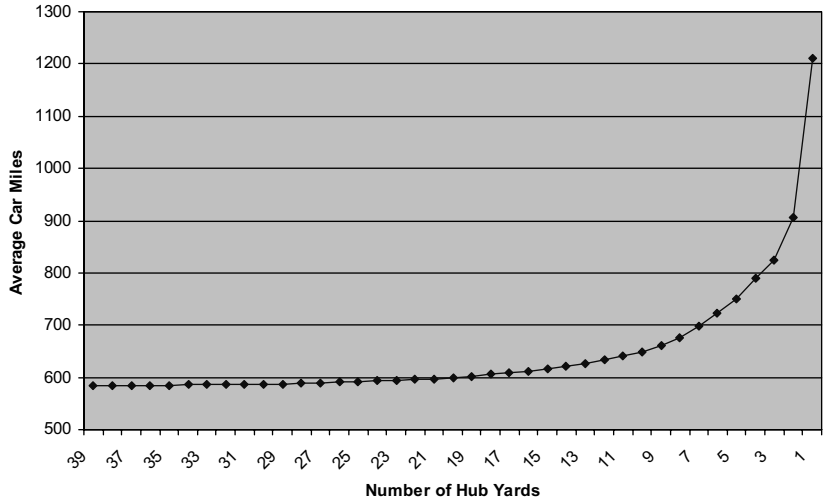
(ii) In the add-drop algorithm, we drop p hub yards one by one and add back q hub yards ($p > q$) one by one.

(iii) The pairwise exchange algorithm swaps a yard location with a nonyard location with the maximum saving in costs if any.

The implementations are tested on a data set of a major U.S. railroad that has specified a set of 39 candidate yard locations from which as many as 17 yards may be dropped, as specified by the railroad company. The add-drop algorithm is tested with $p = 2$ and $q = 1$, and $p = 3$ and $q = 2$. To the solutions obtained by this algorithm, we apply the pairwise exchange algorithm to improve the solution. We find that each of these algorithms give the same solution, which implies that the drop algorithm is quite robust. The running time of the drop algorithm is about six hours on a Pentium IV computer, which is actually a very reasonable time for such a huge strategic-level problem that is not solved on a frequent basis.

The results are evaluated based on two optimization criteria of the blocking problem: car miles and intermediate handlings. Charts of Figures 4 and 5 display average car miles and average intermediate handlings versus the number of yards. Apparently, as the number of hub yards decrease, the average car miles increases while the average intermediate handlings decreases (which is always one when there is only one hub yard in the system). The effect of working with more hubs is almost smooth for the average intermediate handlings. Although

FIGURE 4. Average car miles vs. number of hub yards.



the increase in average car miles is sharp when the number of hubs drops below 10, the change is almost negligible when the number of hubs is between 26 and 39. This shows that this railroad will be able to save a significant amount of money if 10 to 15 hub yards are eliminated, especially when the costs involved in operating yards are considered. When both terms are converted to monetary terms and combined into total transportation costs (see Figure 6), the lowest number of hub yards with almost negligible effect on the total transportation costs goes down to about 18. This analysis supports even more strongly the necessity of eliminating some yards in the system.

To understand the trade-off between deleting yards and the performance of the blocking network, we may combine the two performance criteria into one without even converting them into monetary terms. A general method accepted by the railroads to accomplish this is to find the weighted sum of car miles and intermediate handlings by mapping one unit of intermediate handling to a certain amount of car miles. Table 3 shows the average weighted car miles where one intermediate handling is assumed to be 100, 150, 200, 250, and 300 miles, along with the other figures already displayed in Figures 4, 5, and 6. With this type

FIGURE 5. Average car intermediate handlings vs. number of hub yards.

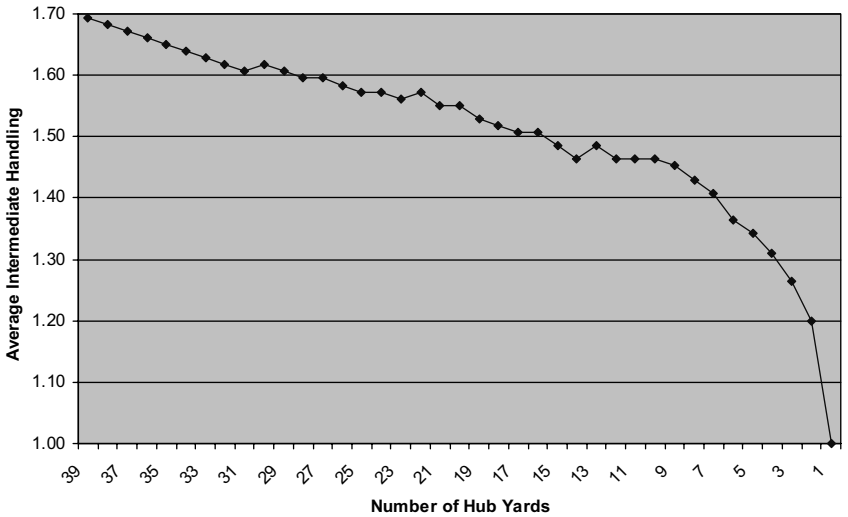


FIGURE 6. Total transportation costs vs. number of hub yards.

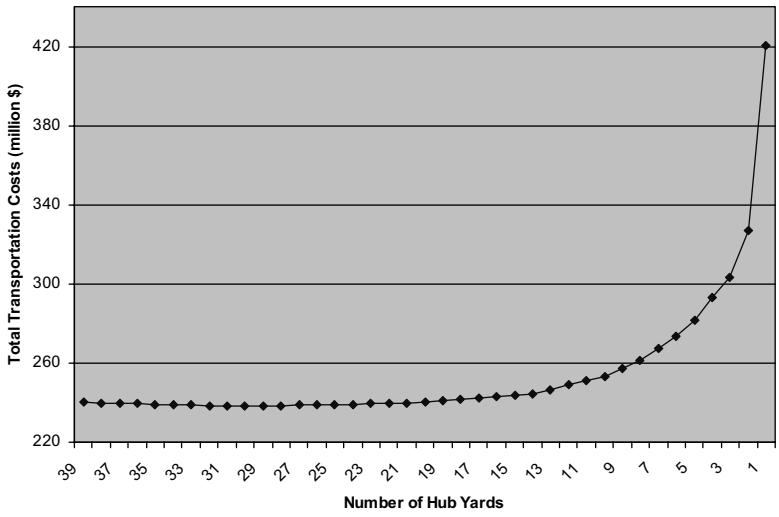


TABLE 3. Results with all performance criteria for different number of hub yards settings.

Number of hubs	Average car miles	Average handling	Average weighted car miles 100	150	200	250	300	Total cost
39	584.00	1.69	753.40	838.10	922.80	1,007.50	1,092.20	240,166,355
38	584.79	1.68	753.09	837.24	921.39	1,005.54	1,089.69	239,756,000
37	584.60	1.67	751.80	835.40	919.00	1,002.60	1,086.20	239,511,525
36	584.73	1.66	750.83	833.88	916.93	999.98	1,083.03	239,300,765
35	585.44	1.65	750.44	832.94	915.44	997.94	1,080.44	239,082,745
34	586.37	1.64	750.27	832.22	914.17	996.12	1,078.07	238,988,145
33	586.26	1.63	749.06	830.46	911.86	993.26	1,074.66	238,664,635
32	585.77	1.62	**747.47**	828.32	909.17	990.02	1,070.87	**238,167,380**
31	587.07	1.61	747.67	**827.97**	908.27	988.57	1,068.87	238,348,055
30	587.25	1.62	748.95	829.80	910.65	991.50	1,072.35	238,502,550
29	587.69	1.61	748.29	828.59	908.89	989.19	1,069.49	238,368,295
28	589.53	1.60	749.03	828.78	908.53	988.28	1,068.03	238,507,940
27	589.97	1.60	749.47	829.22	908.97	988.72	1,068.47	238,725,850
26	591.05	1.58	749.45	828.65	907.85	987.05	1,066.25	238,909,165
25	592.69	1.57	749.99	828.64	**907.29**	985.94	1,064.59	238,984,955
24	593.92	1.57	751.22	829.87	908.52	987.17	1,065.82	239,262,210
23	595.10	1.56	751.30	829.40	907.50	985.60	1,063.70	239,519,170
22	595.43	1.57	752.73	831.38	910.03	988.68	1,067.33	239,903,510
21	597.62	1.55	752.72	830.27	907.82	985.37	1,062.92	239,885,855
20	599.06	1.55	754.16	831.71	909.26	986.81	1,064.36	240,156,015
19	602.29	1.53	755.19	831.64	908.09	**984.54**	1,060.99	240,692,485
18	606.02	1.52	757.82	833.72	909.62	985.52	1,061.42	241,347,150
17	609.69	1.51	760.39	835.74	911.09	986.44	1,061.79	242,185,955
16	611.61	1.51	762.31	837.66	913.01	988.36	1,063.71	242,718,355
15	617.36	1.49	765.86	840.11	914.36	988.61	1,062.86	243,881,330
14	621.59	1.46	767.89	841.04	914.19	987.34	**1,060.49**	244,535,885
13	625.99	1.49	774.49	848.74	922.99	997.24	1,071.49	246,664,000
12	634.33	1.46	780.63	853.78	926.93	1,000.08	1,073.23	248,860,755
11	642.40	1.46	788.70	861.85	935.00	1,008.15	1,081.30	251,240,880
10	649.11	1.46	795.41	868.56	941.71	1,014.86	1,088.01	253,338,030

of analysis, railroads may see the best yard configuration setting (with the number of hubs and their locations) under different cost scenarios even though still not including the costs involved in operating (or even establishing) yards. A boldface number corresponds to the yard configuration with the minimum weighted car miles of its column. For example, if one intermediate handling is equivalent to 300 car miles, then the optimal solution corresponds to having 14 hub yards.

The results show that even if we ignore the savings of closing hub yards, we can always close some hub yards and reduce overall costs. As we close yards, the car miles increase, but intermediate handlings decrease in compensation, and there is very marginal impact to the overall statistics.

Further analysis is to be made by the railroad management by setting appropriate cost terms per car mile and per intermediate handling. Even the analysis without considering the savings due to yard closures reveals a significant amount of savings in total. Hence, it is apparent that the railroads should expect a huge amount of savings by reconsidering the number and locations of their hub yards.

3.4. Concluding Remarks

The yard location problem is one of the topmost strategic-level problems of the railroads, not only due to a high portion of costs involved, but also due to the fact that the blocking network with the yard locations forms the backbone of the transportation operations. Therefore, when the problem is not analyzed from an optimization point of view, the decisions will lead to deficiencies in railroad economies.

To our knowledge, the solution method presented here (although not very complicated once we have an efficient solution method for the blocking problem) is the first attempt in the operations research literature for the railroad yard location problem. Although this strategic problem does not require a real-time decision tool, the solution methods should be simple and fast enough to allow the railroad management to solve and analyze the problem under several scenarios (e.g., yard capacities, different set of candidate yard locations) and for different environmental parameters (e.g., different costs associated with establishing yards or capacity expansions, equivalence of one intermediate handling in car miles). Hence, the railroads should be able to assess such decisions more frequently and be able to respond to the changes in the transportation market and economy.

4. Train Scheduling Problem

Once a railroad has identified a blocking plan, it must design a train schedule so that trains can efficiently carry blocks from their origins to their destinations. The train schedule design problem, henceforth referred to as the *train scheduling* problem, determines: (i) how many trains to run; (ii) the origin, destination, and route of each train; (iii) the train arrival and departure times for each station at which it stops; (iv) the weekly operating schedule for each train; and (v) the assignment of blocks of cars to trains, so that the total cost of transportation (including distance, crew, locomotives, fuel, car hire, etc.) is the minimum possible. These decisions need to be made on a weekly basis, as a train schedule repeats every week.

Major U.S. railroads incur considerable expenses to maintain their operations. One of the largest American railroad companies, BNSF Railway, has approximately 400 long-distance trains. Each train runs an average of five days per week, with an average travel time of one day from its origin to its destination. Hence, a typical train runs about 52 (weeks/year) $\times$ 5 (trains per week) = 260 days. Daily operating costs per train include at least \$1,500 for the locomotive and about \$1,500 for the crew. Thus, the average annual cost to operate a single train is approximately \$0.8 million ($\$3,000 \times 260$ days). An optimization approach that eliminates 10 to 20 trains from the schedule may yield annual savings of \$12–\$20 million. In addition, an optimized train schedule may lead to decreased car flow times, thus reducing car hire cost and improving customer service.

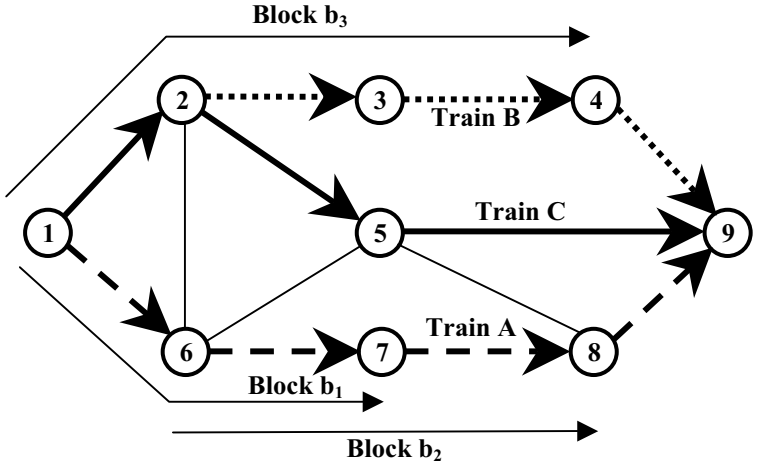
The train scheduling problem is a very large-scale network optimization problem with trillions of decision variables. Its difficulty precluded the development of train scheduling algorithms that are much needed in practical deployment. Some attempts have been made in the past to solve this problem. Early research on train scheduling includes papers by Assad [7], Crainic et al. [21], Crainic and Rosseau [20], Haghani [27], Keaton [31], Keaton [32], and Keaton [33]. These papers divide the train scheduling problem into two separate stages: the train design problem and the block routing problem, which are solved separately. The train design problem determines train routes and the block routing problem routes blocks over the trains formed. An iterative procedure solves each of the two stages in succession, using the solution from the other stage to guide the next iteration. A more recent paper by Gorman [25] considers the integrated train schedule problem, but solves it by a rather crude heuristic. None of these attempts has produced a solution procedure that railroads can use, because they are not scalable for realistically large train scheduling problems, or they ignore the practical realities necessary to generate implementable solutions.

The train scheduling problem comprises three entities: (i) the physical (railroad) network, (ii) trains that travel on the physical network, and (iii) blocks that travel on the trains. These entities are illustrated in Figure 7, which shows a simple example of a railroad network with nine nodes (denoting stations or yards) and the respective train paths and block paths. Train A starts at node 1, follows the route 1-6-7-8-9, and terminates at node 9. Train B follows the route 2-3-4-9, and train C traverses the path 1-2-5-9. Three blocks of cars, b_1, b_2, b_3, travel on these trains. Block b_1 takes train A over the train segments 1-6-7, while block b_2 takes the same train over the segments 6-7-8. So, as a train travels from its origin to its destination, it picks up blocks at various nodes it visits and may also drop off blocks at those nodes. Typically, several blocks ride on a train at any segment. Likewise, a block may travel on several trains as it goes from its origin to its destination. For example, block b_3 starts at node 1 and travels on train C on the segment 1-2. It is off-loaded by train C at node 2, where it is picked up by train B and then carried from node 2 to node 4, its final destination. The transfer of a block from one train to another is called a *block swap*. Among the blocks in our example, only b_3 performs a block swap.

The preceding example provides a brief overview of the structural components of the train scheduling problem. There are five types of decisions to be made in this problem:

(i) What is the number of trains to run?
(ii) What are the origin, destination, and route of each train?
(iii) What are the arrival and departure times at each station that a train visits?

FIGURE 7. Illustrating the interplay between physical network, trains, and blocks.



(iv) What is the frequency of each train? In other words, how often should it run in a week, and on what days?

(v) How should blocks be assigned to trains?

A feasible solution of the train scheduling problem must honor the following constraints:

(i) The number of trains starting at any station within a specified time window (say, every four hours) is limited. This constraint also applies to the number of trains passing through or terminating at a specific station during a given period. It distributes arrivals and departures evenly throughout the day, thereby preventing congestion.

(ii) The number of trains passing through an arc of the physical network is limited. These constraints are called the *line capacity constraints*.

(iii) When blocks are assigned to trains, the number of cars traveling on any train segment cannot exceed a prespecified upper limit. In addition, the number of cars traveling on any segment must be at least a prespecified lower limit. These constraints are called the *maximum and minimum train size constraints*.

The goal of the train scheduling problem is to minimize the weighted sum of the following cost terms: (i) the total distance traveled by cars on trains, (ii) the total flow time of cars as they travel from their origins to their destinations, (iii) the total number of block swaps of cars between trains, (iv) the number of locomotives and crew required, and (v) the total number of trains formed during the week.

In a high-quality train scheduling solution, it is also desirable to minimize the number of train stops at stations, to maintain the consistency of blocks to trains on different days of the week, and to generate crew-friendly and locomotive-friendly train schedules.

The train scheduling problem arises in several areas in a railroad. The version of the problem we discuss here is called *zero-base train scheduling*. In this problem, we determine an entirely new train schedule for a railroad. Railroads typically solve the zero-base train scheduling problem once every several years, when railroad mergers take place or when a major shift in the traffic pattern renders a current train schedule obsolete. However, this task usually has to be performed manually, leaving much room for improvement. If a computerized optimization tool for train scheduling is available, then railroads can do zero-base train scheduling more often—perhaps twice every year. Regular (periodic) changes using the zero-base train scheduling will reduce operational costs and improve efficiency of railroads.

4.1. Mathematical Formulation

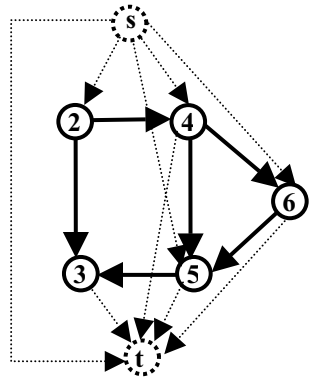
The train scheduling problem is a very large-scale integer programming problem. In this section, we consider a highly simplified version of the problem, in which we assume that each train runs every day of the week (which thus reduces the weekly problem to an equivalent and simpler daily problem), ignore the train timings, and show that even for this case the formulation contains billions of variables and millions of constraints. Figure 8 displays a sample physical network through which trains flow. Let N denote the set of nodes in the network and A denote the set of arcs. We have added a dummy node s to represent train originations, and a dummy node t to represent train terminations. We assume that at source node s, a unit flow of distinct commodities, each one representing a train, is available to be sent to sink node t. A commodity either flows through the network (implying that the corresponding train is formed), or flows through the arc (s, t) (implying that the corresponding train is not formed).

Our formulation features two types of decision variables:

y_a^k : 1 if train $k \in K$ traverses arc $a \in A$, and 0 otherwise; and

x_{ka}^b : 1 if block $b \in B$ flows on arc $a \in A$ of train $k \in K$, and 0 otherwise.

FIGURE 8. Network representation of train paths.



The integer programming formulation for the train scheduling problem is as follows:

$$\text{Minimize} \quad \sum_{k\in K}\sum_{a\in A} f_a^k y_a^k + \sum_{b\in B}\sum_{k\in K}\sum_{a\in A} c_{ka}^b x_{ka}^b + \sum_{b\in B}\sum_{\substack{i\in N \\ i\neq o(b),\, i\neq d(b)}} s_i^b \left| \sum_{a\in O(i)} x_{ka}^b - \sum_{a\in I(i)} x_{ka}^b \right| \tag{19}$$

$$\text{subject to} \quad \sum_{a\in O(i)} y_a^k - \sum_{a\in I(i)} y_a^k = \begin{cases} 1, & \text{if } i = s \\ 0, & \text{if } i \neq s,t \\ -1, & \text{if } i = t \end{cases} \qquad \forall i \in N,\ \forall k \in K \tag{20}$$

$$\sum_{k\in K}\sum_{a=(s,\,i)} y_a^k \le T_i^+, \quad \forall i \in N \tag{21}$$

$$\sum_{k\in K}\sum_{a=(i,\,t)} y_a^k \le T_i^-, \quad \forall i \in N \tag{22}$$

$$\sum_{k\in K} y_k^a \ge T_a, \quad \forall a \in A \tag{23}$$

$$\sum_{k\in K}\sum_{a\in O(i)} x_{ka}^b - \sum_{k\in K}\sum_{a\in I(i)} x_{ka}^b = \begin{cases} 1 & \text{if } i = o(b) \\ 0 & \text{if } i \neq o(b) \text{ or } d(b), \\ -1 & \text{if } i = d(b) \end{cases} \quad \forall i \in N,\ \forall b \in B \tag{24}$$

$$\sum_{b\in B} x_{ka}^b \le M y_a^k, \quad \forall a \in A,\ \forall k \in K \tag{25}$$

$$L_a^k y_a^k \le \sum_{b\in B} v^b x_{ka}^b \le U_a^k y_a^k, \quad \forall k \in K,\ \forall a \in A \tag{26}$$

$$y_a^k \in \{0,1\}, \quad \forall k \in K,\ \forall a \in A \tag{27}$$

$$y_{ka}^b \in \{0,1\}, \quad \forall k \in K,\ \forall a \in A,\ \forall b \in B. \tag{28}$$

In the above formulation, M denotes a large number, and the sets $I(i)$ and $O(i)$, respectively, denote the sets of incoming and outgoing arcs at node $i \in N$. Similarly, $o(b)$ and $d(b)$, respectively, denote the origin and destination of block b. The other notation is self-evident, and we will not describe it here for the sake of brevity. The objective function (19) is the sum of three cost terms: (i) the cost of running trains, (ii) the cost of flowing blocks, and (iii) the cost of block swaps. Constraint (20) states the flow balance constraints for trains.

Constraints (21) (or (22)) state that the number of trains originating (or terminating) at node i per day is no more than T_i^+ (or T_i^-). Constraint (23) captures the restriction that the number of trains traversing arc a per day does not exceed T_a. Constraint (24) states the flow balance constraints for blocks, and (25) ensures that blocks flow on an arc only if a train traverses that arc. Finally, constraints (26) and (27) represent the train-arc design variables and block-to-train assignment variables.

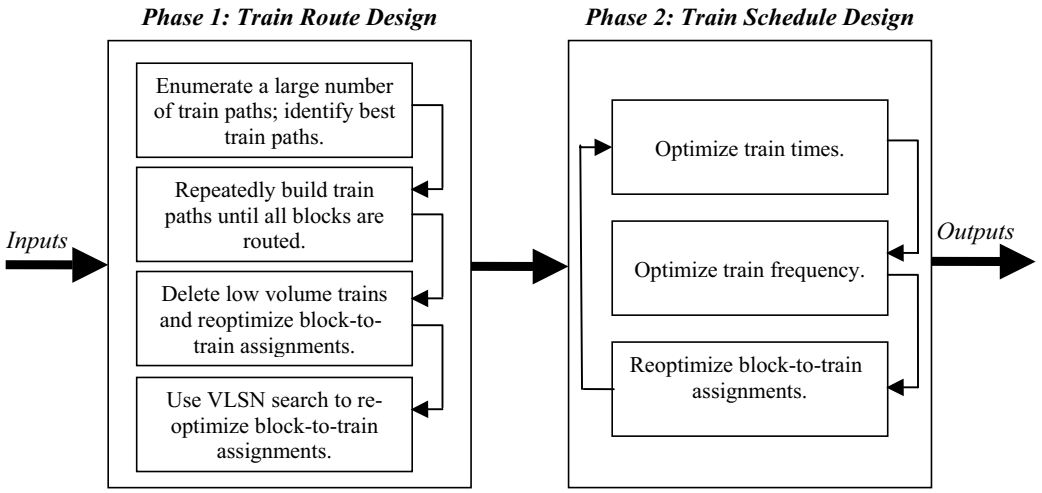
Let us now estimate the size of this IP formulation. For BNSF Railway, the physical network has about 2,000 nodes (where trains can originate, terminate or stop), 12,000 arcs, 1,300 blocks, and 400 trains. For this case, the above formulation contains about 5 million y_a^k binary variables and about 7 billion x_{ka}^b binary variables, as well as approximately 20 million constraints. Integer programming problems of this scale greatly exceed the capacity of existing commercial software.

4.2. Solution Approach

The integer programming formulation demonstrates that the train scheduling problem is a very large-scale discrete optimization problem containing billions of integer variables, even though we ignore some of the critical decision variables, such as train timings and train frequencies. If we consider the weekly scheduling problem instead of the daily scheduling problem, and consider train timings and frequencies, then the resulting integer programming formulation of the train scheduling problem will have trillions of integer decision variables. It is a common approach for problems of this magnitude to be solved by decomposition. In a decomposition approach, instead of determining the optimal values of all decision variables using a single large decision model, we solve a sequence of smaller decision models so that the solutions of these models, when taken collectively, provide a good solution to the integrated model. Decomposition allows us to obtain satisfactory solutions to intractable problems within acceptable running times. However, the method we use to decompose a large problem into a series of smaller problems determines the success of this approach. The approach we discuss here solves the problem in two phases. In the first phase, we determine the train network—i.e., train origins, destinations, and routes as well as block-to-train assignments—that embraces the blocking network; then train frequencies and timings are determined for this train network. The algorithm is summarized in Figure 9.

4.2.1. Phase I: Train Route Design. In this phase, we determine train origins, destinations, and routes, and assign blocks to trains assuming that each train runs every day

FIGURE 9. A schematic representation of the algorithm.



of the week. Our objective is to create a set of daily trains so that all the blocks can be feasibly carried by the train network. The following objective function terms are considered in determining this one-day train schedule:

(i) number of train starts,
(ii) total train miles,
(iii) average number of cars per train mile, and
(iv) number of block swaps.

The algorithm in this phase encompasses two heuristics: construction and improvement. The construction heuristic starts with a null train network and builds trains one by one until all the blocks are routed. The improvement heuristic considers the possible deletions of trains so that the associated objective function terms are improved while there are still sufficient trains to route the blocks. The main steps of these heuristics are summarized next. The steps of the construction heuristic are as follows:

Step 1. Map blocks on the shortest paths in physical network.

Step 2. Starting from each of the block destination, enumerate all the possible trains with the assumption that a train can only originate at a block origin.

Step 3. Attach blocks to possible trains, if the origin and destination of a block is on the route of the train, and a train path exists for a block from its origin to the station where it can take the train under consideration.

Step 4. Determine goodness of each possible train.

Step 5. Create a set of link-disjoint best trains from the set of possible trains.

Step 6. Continue the process till all the blocks are routed.

The steps of the improvement (delete) heuristic are as follows:

Step 1. Consider each train as a candidate for deletion if per-mile car volume is less than a threshold value.

Step 2. Search alternate train route for blocks that are assigned to the candidate train for deletion using the following criteria:

- There should not be more than t trains in new train path for a block.
- A block should not be detoured on a path with distance more than x % of shortest path distance.

Step 3. Delete the candidate train and reroute blocks if alternate train routes are available.

Step 4. Repeat the algorithm until no more deletion in Step 2 is possible.

4.2.2. Phase II: Train Schedule Design. The Phase II algorithm takes as an input the output of Phase I, the train origins, destinations, and routes, and block-to-train assignments; and determines the frequencies and train timings. The algorithm considers a seven-day train space-time network where blocks are routed over the network. We define *train leg* to be a train running on a particular day of the week. For example, if a train runs seven days a week, then it will have seven legs. The network consists of nodes representing physical stops of the trains. The arcs represent the path (more precisely, the path between two consecutive stops) of the trains. At any stop of the train, additional nodes are created to represent stops of individual train legs. These nodes are sorted in the order of departure time of the corresponding train legs. Arcs are introduced to connect these additional nodes in this sorted order.

Phase II consists of three subphases. The first subphase determines the initial train timings and assumes that each train runs every day of the week. The second subphase determines train frequencies, and the third phase reoptimizes train timings by considering the block-to-train assignments.

When we start Phase II, there are no trains added to the space-time network. In the first subphase, we add trains to the space-time network. There are typically two types of trains: those whose times have been specified and those whose times need to be determined. We add the first type of trains and route those blocks that can be completely routed using these types of trains. We next consider the second type of trains and arrange them in the

descending order of the total car miles on them. We then consider each train one by one in this order and determine its time. We use a simple enumeration approach to determine the train timings. We assume that each train runs by the hour, and thus the possible departure times of the train are 00, 01, 02, ..., 23 hours. We also assume that once the departure time of the train has been specified, all other arrival and departure times can be automatically determined. Among these 24 values of the departure time, we consider the departure time for which the total car flow time is minimum.

In the first subphase, we assume that each train runs every day, that is, it has seven legs. However, this may result in some train legs with small volumes. In the second subphase, we determine train frequencies by eliminating such train legs. We repeatedly delete the train legs with the smallest volume and reroute traffic riding on it onto trains on other days. We apply this until each train leg has sufficiently large volume (e.g., at least 25 cars). We consider a particular number of (say 10) train legs with smallest volume, consider each one of them for potential deletion and its impact on the total car flow time, and delete the train leg with the least impact on the total car flow time.

In the second subphase, we change train frequencies, which may make our train timings suboptimal. In the third subphase, we reoptimize train timings by reconsidering the block-to-train assignments. We consider each train one by one in some order and determine its optimal starting time, assuming all other trains do not change their starting times while allowing changes in block assignments. We consider the possible departure times of the train to be 00, 01, 02, ..., 23 hours, and select the time giving the minimum total car flow time.

4.2.3. Incorporating Additional Practical Considerations. Railroads often have some business rules, and they want their plans and schedules to honor those rules. Adherence to these business rules is essential, for otherwise these schedules will not be implemented. To satisfy these business rules and practices, we incorporate the following features in our algorithms:

(i) some trains must or must not be built;
(ii) some train segments or train routes may be given additional preference;
(iii) some train segments or train routes may be undesirable;
(iv) some blocks or traffic may be preassigned to some trains; and
(v) there may be restrictions on a shipment's value, height, and weight that trains must honor.

These features can easily be accommodated by the above heuristics.

4.3. Computational Results

The train scheduling algorithm is implemented to a data set of Norfolk Southern. The results for the performance criteria specified by the railroad are shown in Table 4. The improvements are calculated with respect to the railroad's current solution. The most significant outcome of these results is that the algorithm improves the solution for all criteria without having to consider the trade-offs based on the conflicts among these criteria. In addition, such improvement figures are found by the railroad to be very promising.

Table 4. Results of the algorithm compared to railroad's current solution.

Statistics	% Improvement
Number of train starts	20.2
Total train miles	12.4
Total train travel time	15.6
Average car miles	1.0
Average car days	1.8
Average car block swap	12.7

4.4. Concluding Remarks

Train scheduling is one of the most complex problems among the ones we discuss in this chapter. It is not only the size of the problem, but also integration of several issues that makes the problem complicated. The problem is involved with various decisions that are interrelated with each other, which increases the difficulty in modeling. In addition, an applicable solution targets several objectives to be achieved simultaneously, some of which are conflicting with each other, which increases the difficulty in solving the problem.

In the past, railroads have never been able to approach this problem from an optimization point of view because previous attempts brought poor-quality solutions. Solutions they produce cannot be implemented because they ignore some of the practical constraints. The algorithm we present here not only considers all realistic constraints, but also improves various performance criteria simultaneously. Solutions presented here evidently show that the railroads can improve their train schedules significantly and hence achieve substantial savings in their capital costs by running far fewer trains, and in operational costs by running these trains on shorter routes and with even less frequent block swaps.

5. The Locomotive Scheduling Problem

The *locomotive scheduling* problem is to optimally assign a set of locomotives (also known as a *consist*) to cover all scheduled trains in a given time horizon and to route all these locomotives over the network, while satisfying fleet-size constraints on different locomotive types, fueling and maintenance requirements of the locomotives, pulling-power requirements of all trains, and compatibility restrictions between locomotives and trains. Trains may have different weekly frequencies, i.e., some trains may run every day, while others may run less frequently. The overall aim is to improve the average locomotive utilization and thus reduce the number of locomotives needed.

Locomotive scheduling can be studied at two levels: planning or operational. At the *planning level*, we need to decide the set of locomotives to be assigned to each of the various trains considering the different types available (such as SD40, SD50, AC44), with different horsepower, pulling, and cost characteristics. Some locomotives may *deadhead* on trains, i.e., be pulled like railcars by a set of active locomotives from one place to another, instead of pulling the train. Deadheading allows extra locomotives to be moved as part of a train from locations where they are in surplus to other locations where they are in short supply. Locomotives also *light travel*, i.e., they travel on their own (i.e., not pulling any train or railcars) between different stations to reposition themselves between two successive assignments to trains. Light travel is different from deadheading because it is not limited by the train schedule; in addition, trains travel faster than deadheading. However, light travel is more costly, as a crew is required to operate the light-traveling locomotives, and this movement does not generate any revenue because no cars are attached.

At the *operational level*, we are also concerned with which specific units of the available fleet get assigned to each train. In other words, we need to assign locomotive *tail numbers* (which uniquely identify each individual locomotive) to trains. To do so, we need to take into account the fueling requirements of locomotives. For example, a locomotive may have enough fuel to travel 300 more miles, but a candidate train will not encounter any fueling station for the next 400 miles; clearly, we cannot assign this locomotive to the train. Another important issue is maintenance. Each locomotive needs maintenance at periodic intervals; for instance, regular maintenance every 3,000 miles and major maintenance every 10,000 miles. This maintenance can be done only at some specific locations, called *shops*. Thus, if a locomotive is due for maintenance, then we cannot assign it to a train that takes it too far from a shop, as it cannot return before its due maintenance. Another factor that might be considered in locomotive scheduling at the operational level is uncertainty in train times. Contrary to the planning level, in which it is assumed that all trains run on time, at the

operational level we take into consideration that, in practice, trains are often late. How to account for delayed trains and update locomotive assignments accordingly, with minimum cost impact to the locomotive-related costs, is an important issue that must be considered in an operational-level model.

To assign the available locomotives to trains, we need to account for *consist-busting*. Whenever a train arrives at its destination, its consist (i.e., its set of locomotives) is either assigned to an outbound train in its entirety, or it goes to the pool of locomotives where new consists are formed. In the first case, a *train-to-train connection* occurs between the inbound and the outbound trains, and no consist-busting takes place. In the second case, i.e., some one or more locomotives of a consist are assigned to a departing train while others are assigned to another departing train, consist-busting takes place. Consist-busting entails merging locomotives from inbound trains and regrouping them to make new consists for outbound trains. This may be undesirable for railroads for several reasons: First, it requires additional locomotive and crew time to execute the more complex and time-consuming moves, which requires decoupling and moving locomotives individually; second, it often results in outbound trains getting their locomotives from several inbound trains. If any of these inbound trains is delayed, the outbound train is also delayed, which may lead to further delays down the line. In an ideal schedule we try to maximize the train-to-train connections of locomotives, and thus minimize consist-busting. It should also be noticed that assigning a single, but more powerful, locomotive to a train is undesirable because if that locomotive breaks down, the train gets stranded on the track and blocks the movement of other trains.

Locomotive scheduling is among the most important problems in railroad scheduling, as it involves a very expensive asset. A lack of a planning and scheduling tool for this problem may result in a highly inefficient utilization of this resource. For example, CSX Transportation, a major U.S. railroad, has a fleet of about 3,600 locomotives. This translates into a capital investment of over $6 billion, in addition to over $700 million in yearly maintenance and operational costs. We have observed that the average utilization of locomotives (i.e., when they are actually pulling trains) is around 50%; at other times the locomotives are idling, deadheading, or light traveling. An improvement of 5% in average locomotive utilization may reduce the number of locomotives needed to pull the scheduled trains by 400 (from 3,600 to 3,200), which results in a savings of over $100 million per year in operational costs alone.

Due to its importance, there is a fair share of literature devoted to locomotive scheduling problems. The paper Cordeau et al. [19] presents an excellent survey of existing locomotive scheduling models and algorithms for this problem. Most existing models assume that there are multiple locomotive types available for assignment. However, some models assume that a train is assigned only one locomotive; we refer to such models as *single-locomotive assignment models*. Single-locomotive assignment models can be formulated as variants of integer multicommodity flow problems. Some papers on single-locomotive assignment models are due to Fischetti and Toth [22], Forbes et al. [24], and Wright [49]. Single-locomotive assignment models may be appropriate for some European railroad companies, but are not suited for U.S. railroad companies, because most trains are assigned multiple locomotives. We refer to the models in which multiple locomotives can be assigned to a train as *multiple-locomotive assignment models*. These models have been studied by Chih et al. [16], Florian et al. [23], Nou et al. [40], Smith and Sheffi [47], Ziarati et al. [52], and Ziarati et al. [51]. Most of the proposed models concern locomotive assignment for freight trains. There is little work on locomotive assignment for passenger trains (Cordeau et al. [18] and Ramani [41]).

Though there exist a few programming packages for locomotive scheduling problems, none of them are satisfactory to U.S. railroad companies. Some of the main reasons why existing models do not produce good usable schedules are that: (i) they assign locomotives, not consists, to the trains; (ii) they do not handle deadheading and light travel of locomotives

effectively; and (iii) they do not consider the constraints related to different levels of desirability or preferable assignments. In addition, the current models often produce solutions with excessive consist-busting. This causes delays and introduces costs that are not reflected in the objective function of these models.

5.1. Mathematical Formulation

We focus on the planning version of the locomotive scheduling model (LSM), in which we are to assign a set of locomotives to each train in a weekly train schedule, so that each train receives sufficient pulling power, horsepower, and speed. Delays, fueling, and maintenance needs are not considered. We aim to determine the active and deadheaded locomotives for each train, light-traveling locomotives, and the train-to-train connections, while minimizing the total costs (ownership, active, deadheaded, and light traveling of locomotives, penalty for single-locomotive assignments, and consist-busting).

We are given a weekly train schedule in terms of train routes and times; the set of all trains is given by L. We assume that the same train running on different days is represented by different trains in the set L, one for each day it runs. For each train $l \in L$, we are given its origin and destination stations, as well as the respective departure and arrival times. Let T_l and β_l be the tonnage requirement and the horsepower per tonnage needed for train l, and E_l be the penalty for using a single-locomotive consist.

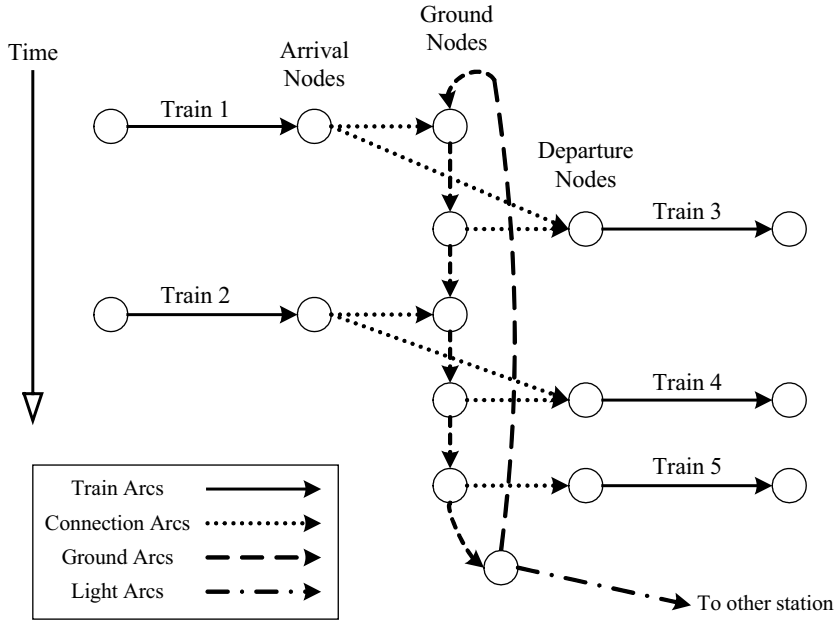
We are also given a set K of all locomotive types that are available for pulling the trains. Let h^k and λ^k represent, respectively, the horsepower and the number of axles for each locomotive type $k \in K$. The weekly ownership cost of a locomotive of type k is given by G^k, while B^k represents its fleet size, i.e., the respective number of locomotives available for assignment. Each active locomotive of type k assigned to train l provides the tonnage pulling capability t_l^k.

We formulate the LSM as an integer multicommodity flow problem with side constraints. This mixed-integer programming (MIP) formulation represents the flow of locomotives on a *weekly space-time network*. In this network, arcs denote trains; nodes denote events (that is, arrivals and departures of trains and locomotives); and each different locomotive type defines a commodity.

We denote the weekly space-time network as $G = (N, A)$, where N and A represent the node and the arc sets, respectively. For each arrival event, we create an *arrival ground node*, and for each departure event we create a *departure ground* node. Each node $i \in N$ is associated with two attributes: place and time. The sets of nodes related to train departures and arrivals are denoted, respectively, by *DepNodes* and *ArrNodes*, and the ground nodes by *GrNodes*. Let $AllNodes = DepNodes \cup ArrNodes \cup GrNodes$.

This network contains four types of arcs, as shown in Figure 10. The set of *train arcs* (*TrArcs*) contains one arc for every scheduled train; its endpoints denote the location and time of the departure and arrival events. We connect each arrival node to the associated arrival-ground node by a directed arc called *arrival-ground connection arc*; similarly, we connect each departure-ground node to the associated departure node through a directed arc called *ground-departure connection arc*. These connection arcs belong to the set of connection arcs (*CoArcs*). We next sort all ground nodes at each station in chronological order by their time attributes and connect each ground node to the next ground node through directed arcs called *ground arcs*, denoted by the set *GrArcs* (we assume without any loss of generality that ground nodes at each station have distinct time attributes). These ground arcs allow inbound locomotives to stay in an inventory pool as they wait to be connected to the outbound trains. An inbound train that sends its entire consist to an outbound train is represented by a *train-to-train connection arc*, which directly links an arrival node to a departure node; these arcs also belong to the set *CoArcs*. Finally, we also allow the possibility of locomotives light travel. We create a *light arc* originating at a ground node (with a specific time and at a specific station) and also terminating at a ground node. Each light arc belongs to

FIGURE 10. Part of a weekly space-time network for a station.



the set *LiArcs* and has a fixed charge F_l that denotes the fixed cost of sending a single locomotive with crew from the origin of the light arc to its destination. The light arc also has a variable cost that depends on the number of locomotives light traveling as a group. Let $AllArcs = TrArcs \cup CoArcs \cup GrArcs \cup LiArcs$, and d_l^k be the cost of deadheading of locomotive type k on a train arc $l \in TrArcs$; for every other arc, d_l^k denotes the cost of traveling for a nonactive locomotive of type k on arc. For each node i, the sets of incoming arcs into i and outgoing arcs from i are given by $I(i)$ and $O(i)$, respectively.

Our formulation has five sets of decision variables: (i) x_l^k is the number of active locomotives of type k allocated to train l; (ii) y_l^k is the number of nonactive locomotives (deadheading, light-traveling, or idling) of type k on all arcs; (iii) z_l equals 1 if at least one locomotive flows on the arc $l \in LiArcs \cup CoArcs$, and 0 otherwise; (iv) w_l equals 1 a single locomotive is assigned to train l, and 0 otherwise; and (v) s^k indicates the number of unused locomotives of type k. The MIP formulation of the LSM is given by:

$$\text{Minimize} \quad \sum_{l \in TrArcs} \sum_{k \in K} c_l^k x_l^k + \sum_{l \in AllArcs} \sum_{k \in K} d_l^k y_l^k + \sum_{l \in LiArcs} F_l z_l + \sum_{l \in CB} V z_l + \sum_{l \in TrArcs} E_l w_l - \sum_{k \in K} G^k s^k \tag{29}$$

$$\text{subject to} \quad \sum_{k \in K} t_l^k x_l^k \geq T_l \qquad \forall\, l \in TrArcs \tag{30}$$

$$\sum_{k \in K} h_l^k x_l^k \geq \beta_l T_l \quad \forall\, l \in TrArcs \tag{31}$$

$$\sum_{k \in K} \lambda_l^k x_l^k \leq 24 \quad \forall\, l \in TrArcs \tag{32}$$

$$\sum_{k \in K} (x_l^k + y_l^k) \leq 12 \quad \forall\, l \in TrArcs \cup LiArcs \tag{33}$$

$$\sum_{l \in I(i)} (x_l^k + y_l^k) = \sum_{l \in O(i)} (x_l^k + y_l^k) \quad \forall\, i \in AllNodes,\ \forall\, k \in K \tag{34}$$

$$\sum_{k\in K} y_l^k \le 12 z_l \quad \forall\, l \in CoArcs \cup LiArcs \tag{35}$$

$$\sum_{l\in O(i)} z_l = 1 \quad \forall\, i \in ArrNodes \tag{36}$$

$$\sum_{l\in I(i)} z_l = 1 \quad \forall\, i \in DepNodes \tag{37}$$

$$\sum_{k\in K} (x_l^k + y_l^k) + w_l \ge 2 \quad \forall\, l \in TrArcs \tag{38}$$

$$\sum_{l\in S} (x_l^k + y_l^k) + s^k = B^k \quad \forall\, k \in K \tag{39}$$

$$x_l^k \in \mathrm{Z}^+ \quad \forall\, l \in TrArcs,\ \forall\, k \in K \tag{40}$$

$$y_l^k \in \mathrm{Z}^+ \quad \forall\, l \in TrArcs,\ \forall\, k \in K \tag{41}$$

$$z_l \in \{0,1\} \quad \forall\, l \in CoArcs \cup LiArcs \tag{42}$$

$$w_l \in \{0,1\} \quad \forall\, l \in TrArcs. \tag{43}$$

The objective function (29) contains six terms. The first term denotes the cost c_l^k of actively pulling locomotives on train arcs. The second term captures the cost d_l^k of deadheading locomotives on both train arcs and light travel arcs, as well as the cost of idling locomotives. The third term denotes the fixed cost F_l of light-traveling locomotives. The fourth term denotes the fixed cost V of consist-busting for the set CB of all connection arcs linking arrival nodes to ground nodes. The fifth term denotes the penalty E_l associated with single-locomotive consists, while the sixth term represents the savings G^k accrued from not using all the available locomotives. Observe that we can obtain different levels of consist-busting by selecting different values of the consist-busting penalty cost V. The greater the value of V, the less the amount of consist-busting is in the optimal solution.

Constraints (30) and (31) ensure that the locomotives assigned to a train provide the required tonnage and horsepower, respectively. Constraint (32) enforces the maximum number of active axles assigned to a train, while constraint (33) ensures that each train arc and light arc is assigned at most 12 locomotives. Constraint (34) relates to the flow balance (number of incoming and outgoing locomotives of each type k) at every node. Constraint (35) captures the occurrence of a positive flow on a connection arc or a light arc; it also ensures that no more than 12 locomotives flow on any light arc. Constraints (36) and (37) ensure that either a train-to-train connection or a consist-busting occurs for each inbound and outbound train. The assignment of a single-locomotive consist to a train is represented by constraint (38). Finally, constraint (39) accounts for the total number of locomotives used in the week, given by the sum of the flow of locomotives on all the arcs crossing a time instant of the week when no event (arrival or departure of train) takes place, denoted by *CheckTime*; the difference between the number of locomotives available and the number of locomotives used equals the number of unused locomotives (s^k).

The locomotive scheduling problem is a very large combinatorial optimization problem and is also NP-complete (Ahuja et al. [3]). A large railroad has a train schedule that consists of several hundred trains with different weekly frequencies, which translates into several thousand train departures per week. Some trains run every day in a week, whereas others run less frequently. Many trains have long distances to travel and take several days to go from their origins to their destinations. To power these trains, several thousand locomotives of different types are available, with different costs, horsepower, and pulling capacities. To illustrate this, for a real-world problem, with 538 trains (each of which operated several days in a week) and 5 locomotive types, the size of the resulting formulation contains about 197,000 integer variables and 67,000 constraints. This problem is too large to be solved to optimality or near-optimality using existing commercial-level MIP software.

5.2. Solution Method

It should be noted that the above mathematical formulation (29)–(42) does not include other constraints found in practice. A railroad may want a solution that is *consistent* throughout the week in terms of the locomotive assignment and train-to-train connections. If a train runs five days a week, we want it to be assigned the same consist each day it runs. Consistency of the locomotive assignments and train-to-train connections are highly desirable from an operational point of view, because it translates into rule-based dispatching that makes the job of a locomotive dispatcher relatively easy (the dispatcher can remember the rule and enforce it). In addition, assignments of locomotive types to trains may have different degrees of preference.

In order to cope with these additional consistency constraints, which would make the formulation even more complex in terms of additional variables and constraints to be added, not to mention that the network size would be increased dramatically, Ahuja et al. [6] propose an efficient decomposition-based heuristic approach that allows near-optimal solutions for real instances of the LSM to be obtained in short to moderate computing times. This heuristic, which makes extensive use of the commercial software CPLEX 7.0, comprises two main stages:

(i) Modify the original problem so that all trains run seven days a week, and then solve the daily locomotive scheduling problem instead of the weekly scheduling problem.

(ii) Adjust the solution of the first stage (daily schedule) to solve the original weekly schedule problem in which trains do not all run seven days a week.

5.2.1. Solving the Daily Locomotive Scheduling Problem. This first stage of the solution approach is a simplification that reduces the weekly locomotive problem to a daily problem, thus reducing the size of the MIP substantially, and also helps satisfy the consistency constraints. A *daily space-time* network is created similarly to the weekly network described above and is about seven times smaller. In the simplified model it is assumed that (i) all trains that run p days or more per week run *every* day of the week, and (ii) all trains that run less than p days do not run at all. This assumption results in an approximation in the sense that it provides locomotives to trains that do not actually exist; similarly, it may not provide locomotives to trains that do exist. This simplification is based on the observation of real data, for which the vast majority of train arcs in the weekly space-time network correspond to the trains that run at least five days a week. Some minor modification is required in the mathematical formulation in order to account for the adjusted fleet size for the daily problem.

Though the daily space-time network is substantially smaller, it is still too large to be solved to optimality using commercial MIP software. To illustrate this, the daily space-time network equivalent to the weekly network described above contains 1,323 nodes and 30,034 arcs. Though the LP relaxation takes only a few seconds to solve, our experiments show that the MIP does not provide any feasible integer solution in 72 hours of running time. We conjecture that the biggest source of difficulty is the presence of the fixed-charge variables z_l (for connection and light arcs). In this manner, to obtain high-quality feasible solutions and to maintain a relatively small total running time of the algorithm, the idea is to eliminate these fixed-charge variables using the following three-step sequential heuristic approach:

Step 1: *Determine Train-to-Train Connections.* In this step we consider that railroads usually specify some train-to-train connections, thus allowing us to eliminate inadmissible train-to-train connections. This can be done heuristically in the following way: (i) add all the candidate train-to-train connections; (ii) fix these train-to-train connections one by one and solve the LP relaxation of the daily MIP formulation to assess the impact of these connections; and (iii) keep those connections with small impact on the cost of assignment.

Step 2: *Determine Light Movements.* To allow light travel of locomotives from any station to any other station at any time of the day, it is necessary to add a large number of arcs

in the daily space-time network. Instead, we use heuristic procedures to create a small but potentially useful collection of light-travel arcs, and to select a subset of these arcs for light travel. To achieve this, we (i) add all the candidate light arcs, (ii) remove the candidate light arcs one by one and assess the impact of these removals using the LP relaxation of the MIP formulation, and (iii) remove the light movements with minimal impact on the cost of flow.

Step 3: *Determine the Assignment of Locomotives.* Once the fixed-charge variables are eliminated through the previous steps, we determine the remaining variables, which correspond to the arcs related to the movement of the active and deadheaded locomotives. To accomplish this, we solve the integer program for the daily locomotive assignment without the fixed-charge consist-busting and light-travel variables. Though CPLEX can provide a high-quality solution within 15 minutes of execution time, it does not terminate even when it is allowed to run for over 48 hours. On the other hand, this initial feasible solution can easily be improved by a modest modification. Thus, we use a neighborhood search algorithm to look for possible improvements with respect to this solution.

The above steps allow us to reach a near-optimal solution to the daily locomotive scheduling problem in short computing times.

5.2.2. Solving the Weekly Locomotive Scheduling Problem. When the daily scheduling solution is applied to the corresponding weekly scheduling problem (by repeating it every day of the week), some trains may not be assigned any consist (those that run less than p days in a week) and some locomotives may be assigned to trains that do not exist (those that do not run every day). To transform this daily solution into a feasible solution for the original weekly problem, locomotives are taken from the trains that exist in the daily problem but do not exist in the weekly problem and assigned to the trains that do not exist in the daily problem but exist in the weekly problem. It should be noted that additional locomotives may eventually be required to meet these constraints.

A modified weekly space-time network is then constructed in the same manner as the daily space-time network, where each train running on different days is treated as a separate train. However, we make some changes for the train-to-train connection arcs. For a train-to-train connection arc in the daily space-time network, we have a corresponding train-to-train connection arc only on those days when both the associated trains run, and not otherwise.

Due to the size of the resulting modified MIP flow formulation that solves the weekly problem that is based on the solution of the daily problem, and thus requires excessive computing time, a heuristic approach is applied in which the locomotive schedule is determined for one locomotive type at a time. This can be accomplished by converting the multicommodity flow problem (with each locomotive type defining a commodity) into a sequence of single commodity flow problems with side constraints, one for each locomotive type. The aim is to determine the optimal flow of the current selected locomotive type while keeping the flow of other locomotive types intact, similar to the approach used for the daily scheduling problem. To do so, the different locomotive types are first arranged in decreasing order of their availability and then, following this order, considered one by one. Each of these subproblems is a (single-commodity) constrained minimum cost flow problem and can be solved very efficiently. After a feasible integer solution of the weekly locomotive scheduling problem is obtained, a neighborhood search algorithm is applied to improve it. The search is terminated when the solution value does not improve for any locomotive type. Finally, after the solution of the weekly locomotive scheduling problem is obtained, additional train-to-train connections are created by matching the locomotive assignments of inbound and outbound trains at each station if they have the same consist.

To summarize, the methodology for solving the planning version of the locomotive scheduling problem involves the following main steps: (i) modeling the weekly scheduling problem as a MIP multicommodity flow problem with side constraints; (ii) transforming this problem into a daily scheduling problem; (iii) heuristically solving a simplified version of the daily

Table 5. Comparison of heuristic algorithm and CSX solutions.

Locomotive type	Proposed heuristic	CSX solution	Measure	Proposed heuristic (%)	CSX solution (%)
SD40	249	498	Active time	44.4	31.3
SD50	160	171	Deadheading time	8.1	19.6
C40	487	621	Idling time	46.7	49.1
AC44	154	164	Light-traveling time	0.8	0
AC60	160	160	Consist-busting	49.4	85.0
Total	1210	1614			

scheduling problem (in which some fixed-charge variables are previously set through greedy algorithms); and (iv) transforming the daily solution into a feasible and effective solution for the weekly problem.

5.3. Computational Results

In Table 5, we present the comparison of the results obtained by Ahuja et al. [6] with the ones obtained by the software developed in-house by CSX, a major U.S. railroad company. The algorithms were implemented in C++ and run and tested on a Pentium III PC with a speed of 750 MHz. The results indicate that whereas CSX software required 1,614 locomotives to satisfy the train schedule, the heuristic required only 1,210 locomotives, thereby achieving a savings of over 400 locomotives, or about 25% of the fleet. The locomotive utilization increased by 13.1%, while the deadhead and idling times decreased by 9.5% and 2.4%, respectively. The authors report similar improvements on other benchmark instances. We thus observe that this heuristic produces far superior solutions and demonstrates major savings potential.

5.4. Concluding Remarks

The results obtained so far by the heuristic algorithm to solve the planning version of the locomotive scheduling problem produces solutions that satisfy the constraints and business rules usually required by railroads, and offers considerable savings in cost, especially in terms of a significant reduction in the number of locomotives needed. Results are obtained in very short running times, usually less than one hour. The model is the first optimization approach to account for consist-busting, light travel, and consistency of the solution in a unified framework.

On the other hand, there is a fair amount of additional research that needs to be done before an optimization model can be used by railroads in their day-to-day locomotive planning and scheduling. It should consider the constraints that appear in the operational level, including the assignment of specific units of each locomotive type to each train, the fueling and maintenance needs of the locomotives, and train deviations from their scheduled times. Another area of potential interest for research is the assignment of locomotives for passenger trains, which may differ quite substantially from freight train problems in terms of its objective and constraints, though the ideas and algorithms presented here may be useful and applicable.

6. Train Dispatching Problem

The *train dispatching* problem, also known as the *train meet-and-pass* problem, aims to determine detailed train movements and timetables over a rail network under various operational constraints and restrictions in order to minimize train deviations from the planned schedule. This problem assumes that an ideal train schedule is given—i.e., for each scheduled train we know the planned departure and arrival times at the corresponding origin and

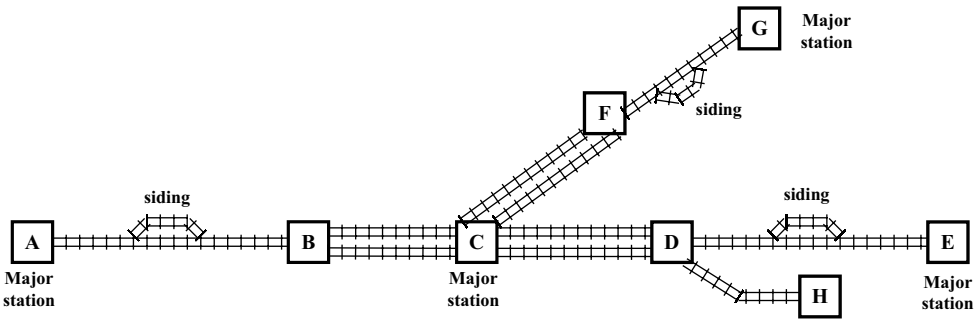
destination stations, as well as the travel times along the portions of its itinerary. This ideal train schedule results from the solution of the train scheduling problem that we discussed in §4. In the train scheduling problem we assign blocks to trains and identify a master schedule in terms of train paths and timings with respect to main stations and origin-destination points of the shipments. However, this master schedule does not consider the conflicts that occur as the trains meet along the tracks between these main stations.

This problem is particularly relevant for most freight railroads, ones that usually operate over extensive lengths of single-line tracks, in which a train has to be pulled over to cross or overtake another train, causing undesirable delays to freight and shipments, not to mention reductions in the productivity of locomotives and railcars. The related rail network may be composed of one or more track lines connecting major stations. Each track line may encompass several intermediate single- and double-line sections between intermediate stations (terminals or junction points). A single-line section is defined as a segment of track line that can accommodate only one train at a time. Similarly, a double-line section can accommodate up to two trains at a time, traveling either in the same direction or in opposite directions. A track section is delimited at each of its extreme endpoints by a siding or an intermediate station. A siding can accommodate at least one train waiting for crossing or overtaking, while an intermediate station serves as a yard or a terminal where passing trains can wait for crossing or overtaking and from which trains can depart or arrive. An example for a partial rail network is given in Figure 11, where stations A, C, E, and G can be considered as major stations. Track sections B-C, C-D, and C-F are double line. Sidings are located in sections A-B, D-E, and F-G.

The detailed dispatching plan consists of a timetable that comprises the departure and arrival times of each train in each track section of its route in such a way that all rail traffic constraints are met, especially track capacities, thus ensuring that all train crossings and overtakes happen at passing sidings or intermediate stations where there is enough room for the involved trains to be accommodated. Therefore, the resulting dispatching plan also encompasses the detailed information of all trains that are delayed due to conflicts with other trains, such as their locations and the start and end times of each nonservice stop where trains are pulled over solely for crossing or overtaking.

In addition to operational constraints that honor the rail traffic movement, some other practical constraints may arise. Insufficient track length or the nature of the freight being carried may restrict the trains that can be stopped at a location (e.g., a train of hazardous products may not be allowed to pull over near a populated area). Some trains may not be allowed to stop at some specific sidings because they may not have enough traction power to resume moving if they are pulled over, depending on the specific track slope ahead of the siding towards the direction the train is to move. Similarly, the lack of adequate infrastructure for crews may limit the maximum time a train can wait at a siding. Maximum allowed delays for individual trains and train types play a critical role in attaining consistent

FIGURE 11. Part of a rail network with single and double-line tracks.



schedules and dispatching plans. Other constraints found in practice may include meeting exact timetables for some trains, minimum and maximum departure and arrival headways for trains of the same type following the same route (to avoid unnecessary wait time by arriving too early to load or unload), and ensuring that these trains arrive on time and in the same sequence as they departed.

The train dispatching problem may arise in different contexts. In real-time scheduling, it aims to help the dispatchers who are controlling the traffic to make sound decisions about which trains to stop and where, as updated data about train positions become available. In tactical planning, it consists of finding the best master timetable schedule on a regular periodic basis (e.g., weekly, monthly, or quarterly). In strategic planning, it relates to investment decisions in railroad infrastructure like new sidings to be built, extension of current sidings to accommodate longer trains, and duplication of single-line segments. This problem has been receiving increasing attention lately as advanced control systems that provide real-time information on train position and velocity are being developed and implemented by several railroad companies.

This problem is a well-studied railroad optimization problem in the operations research literature. Several earlier research papers study analytical line models whose main aim is to estimate the delay to each train caused by interferences on a rail line as a function of dispatching policies, traffic distribution, and physical track topology. A good summary review of these papers can be found in Cordeau et al. [19]. Starting in the mid 80s, computerized train dispatching tools have been proposed. The papers Rivier and Tzieropoulos [42], Rivier and Tzieropoulos [43] and Sauder and Westerman [46] contain these studies. Particularly for European railroads that mostly run passenger trains, there are several studies that consider dispatching and track allocation problems (see Adenso-Diaz et al. [1], Carey and Lockwood [14], Kroon and Peeters [35]). Integer programming formulations are proposed in Brännlund et al. [10]; Caprara et al. [12]; Higgins et al. [28, 29]; and Kraay and Harker [34], as well as solution methods to solve these problems approximately. As the size of the problem increases, thus making it difficult to solve to optimality, not to mention the necessity of a tool to reach a solution quickly (specifically for real-time dispatching decisions), a number of heuristic algorithms have also been proposed (see Cai et al. [11] and Şahin [44]).

However, the algorithms developed so far rely on simplifications of the general problem to solve specific instances. Several papers deal with problems related to passenger trains, in which the objective function and the constraints may differ quite significantly from freight trains. The intractability of the formulations for freight train dispatching, some modeled even as nonlinear optimization problems, have led to simplifications to allow finding a feasible solution that sometimes is not even known to be applicable in practice. In general, the problems consider a railroad line linking two major points. None of them consider explicitly a rail network composed of different track lines and the interactions between different trains traveling from different origins to different destinations that share distinct parts of their routes. Decomposition approaches for such networks may lead to good, near-optimal partial solutions for each track line, but often result in poor global solutions due to the intrinsically greedy nature of this type of approach, which does not consider the impact of a train crossing the boundaries of a partial network on the traffic of the other trains in the entering portion of the network. Specific operational constraints such as those related to train size and type as mentioned before are considered neither in the formulations nor in the solutions proposed.

6.1. Modeling Approach

The paper in Şahin et al. [45] proposes a novel approach for modeling the train dispatching problem as a multicommodity flow problem with side constraints on a space-time network. Each train t (belonging to a set T of trains to be dispatched) functions as a commodity in the network. The space-time network representation allows the formulation to be easily adapted for intersecting complex rail networks instead of only a single major line, as well

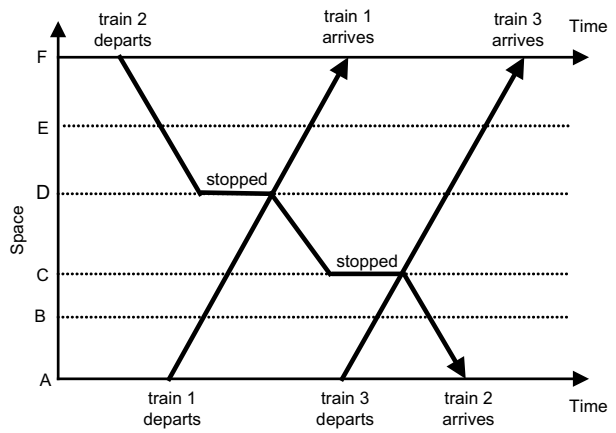
as for multitrack lines, as shown before in Figure 11. Most of the practical constraints that would make the problem size even larger with the previous formulation approaches found in the literature are handled within the network construction phase. The proposed formulation is flexible enough to incorporate most of the practical constraints without having to change its underlying structure.

6.1.1. Network Representation. For the sake of clarity on how the space-time network is built, we will consider a track line that links two major stations and is composed of single- or double-track sections; the generalization to a more complex structure involving multiple interconnecting lines is straightforward and requires no major change. The proposed network representation is based on the concept of a train graph diagram, which is a standard method for displaying the resulting train schedules, meet-points, and associated delays. Many freight railroads still rely on train schedules that are made manually by skilled dispatchers using this kind of diagram. Figure 12 illustrates a sample train graph schedule diagram for a single-line track that links terminal stations A and F. Intermediate meet-points are located at stations B to E. Vertical and horizontal axes represent space and time, respectively. There are two northbound trains that move from A to F, and one southbound train. The southbound train is delayed twice, at meet-points D and C, as a possible solution for the conflicts with the two northbound trains. In other words, these delays allow northbound trains to meet and pass.

Even in this very simple example, there are many possible combinations of sidings and times for trains to be pulled over to allow meets and passes, not to mention eventual changes in the departure times from the intermediate stations. Therefore, the train dispatching problem is a very large-scale combinatorial optimization problem. When several trains in both directions are scheduled, many conflicts may arise. Each conflict involves trains moving either in opposite directions or in the same direction. Depending on the chosen solution for a conflict involving two trains (i.e., which train pulls over for the other to pass), the location and the time of later conflicts may change, new conflicts at different locations and times may arise, and existing conflicts may cease to exist. Thus, the number of possible solutions to train conflicts can be very large.

On the line track linking the two major stations, let $S = \{0, \ldots, s\}$ represent the set of stations and sidings, numbered according to the order in which they appear along the track line (say from north to south). Each train $t \in T$ may depart from any station $s_1 \in S$ and arrive at any other station $s_2 \in S$ $(s_1 \neq s_2)$. We also define the set of sections (line segments) to represent the tracks between consecutive stations or sidings s_i and s_{i+1} as $L = \{0, \ldots, s-1\}$. $IS(t) \subset S$ denotes the set of intermediate stations or sidings $s \in S$ where train $t \in T$ is allowed to pull over to wait for crossing or overtaking while $P(t) \subset L$ denotes the ordered

FIGURE 12. A train graph schedule diagram.

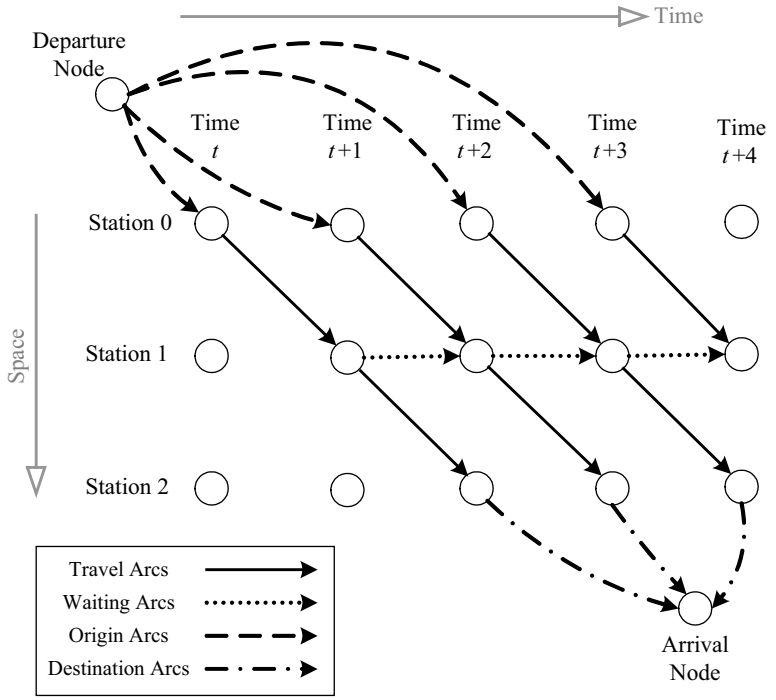


set of consecutive sections $(d^t - s_1, s_1 - s_2, \ldots, s_k - a^t)$ that compose the train path. For each train $t \in T$ we are given the origin (d^t) and destination (a^t) stations; the corresponding scheduled departure time ed^t is known, as well as the travel time along track segment $i(f_i^t)$. The scheduled arrival time at destination (sa^t) can be calculated by $sa^t = ed^t + \sum_{l \in P(t)} f_l^t$. For each train t, we also specify the maximum allowable delay md^t, which is based on business rules or operating principles. Accordingly, latest possible departure time for train t (ld^t) and latest possible arrival time for train $t(la^t)$ can be calculated as $ld^t = ed^t + f_i^t$ and $la^t = sa^t + f_i^t$.

We represent the space-time network as $G = (N, A)$, where N and A denote node and arc sets. The space-time network contains three types of nodes. *DepNodes* represents the set of artificial nodes in which the outflow generates departure of a train from its origin. There exists a departure node for each train $t \in T$ denoted as Dep^t. Therefore, the supply of each node in this set is one. The *ArrNodes* set corresponds to the set of artificial nodes in which the inflow represents the arrival of a train to its final destination. In a similar way, its demand is set to one and denoted as Arr^t, for all $t \in T$. *StatTimeNodes* is the set of nodes with two attributes: station/siding and time. Time is discretized with respect to equal-length time periods. For instance, if a daily schedule (24 hours) is discretized in 10-minutes periods, then $q = 144$ where the time period set is $Q = \{1, 2, \ldots, q\}$. Nodes of the set *StatTimeNodes* correspond to the copies of each station node $s \in S$ in each time period $k \in Q$. An element of this set is denoted as i_k, where $i \in S$ and $k \in Q$.

The arc set A is composed of four subsets as shown in Figure 13. The arcs in the set of *origin arcs* emanate from nodes in *DepNodes*, and *destination arcs* enter the nodes in *ArrNodes*. To represent the departure of a train t from its origin station d^t, we create arcs from node Dep^t to nodes d_k^t for each time period k where $ed^t \le k \le ld^t$. Similarly, to model the arrival of a train t at its destination station a^t, we create arcs from nodes a_k^t to node Arr^t for $sa^t \le k \le q'$, where $q^t = \min\{q, la^t\}$.

FIGURE 13. Part of a space-time network for a train.



The arcs of the space-time network are created on a train-by-train basis following the sequence of stations/sidings $s \in IS(t)$, and segments $l \in P(t)$ on the train's route, starting from its origin station d^t through its destination station a^t. The *travel arcs* for a train are constructed between each pair of consecutive stations on its path for only the time periods that lie within the earliest and latest travel times for the section. The set of *waiting arcs* for train t includes the arcs from node i_k to node i_{k+1}, corresponding to the siding $s \in IS(t)$ where the train is allowed to wait. Whenever i_k has an incoming arc that represents a possible movement of train t, we create consecutive waiting arcs on the same station level by considering the maximum possible delay of train t at that station.

Practical constraints can be considered and thus incorporated into the model by adding new arcs or getting rid of certain arcs on the network. Therefore, for instance, if a train cannot wait at a particular siding, then the corresponding waiting arcs are not created. Another restriction might require that specific trains (typically passenger trains) be stopped at certain intermediate stations during certain time windows. We handle this by adding the waiting arcs and avoiding the associated travel arcs on the network. Other practical constraints are handled similarly during the network construction phase instead of explicitly expressing those constraints in the formulation.

All arc capacities are set to 1. The costs of waiting arcs are set to the length (in minutes) in which time is discretized, because each arc represents a train waiting at a particular siding from time k to time $k+1$. Similarly, costs for origin arcs correspond to the respective lengths of the initial delays. All other arcs' costs equal 0. Delays can also be multiplied by specific train weights to reflect relative priority of certain trains.

6.1.2. Integer Programming Formulation. Our IP formulation has binary flow decision variables $x^t_{i_k j_l}$ representing the flow of train t on arc (i_k, j_l) where $t \in T$ and $(i_k, j_l) \in A$. Note that for a given train t, each binary flow variable is defined only for feasible moves corresponding to one of the four arc sets, as described above.

$$\text{Minimize} \quad \sum_{t \in T} \sum_{(i,j) \in A} c_{ij} x^t_{ij} \tag{44}$$

$$\text{subject to} \quad \sum_{ed^t \le k \le ld^t} x^t_{Dep^t d^t_k} = 1 \quad \forall t \in T \tag{45}$$

$$\sum_{sa^t \le k \le la^t} x^t_{a^t_k Arr^t} = 1 \quad \forall t \in T \tag{46}$$

$$\sum_{i \in StatTimeNodes} x^t_{ij} - \sum_{i \in StatTimeNodes} x^t_{ji} = 0 \quad \forall t \in T,\ \forall j \in StatTimeNodes \tag{47}$$

$$\sum_{t} x^t_{ij} \le 1 \quad \forall (i,j) \in A \tag{48}$$

$$\sum_{t \in T^{out}} \sum_{l \le k-1} \sum_{m \ge k} x^t_{i_l (i+1)_m} + \sum_{t \in T^{in}} \sum_{l \le k-1} \sum_{m \ge k} x^t_{(i+1)_l i_m} \le 1 \quad \forall i \in S,\ \forall k \in Q \tag{49}$$

$$x^t_{ij} \in \{0,1\} \quad \forall (i,j) \in A,\ \forall t \in T. \tag{50}$$

The objective function (44) minimizes the total delay of all trains. Constraint (45) ensures that for each train t there is a unit outflow from its departure node to any node in *StatTimeNode* that corresponds to the origin station at a time within its departure time window. Similarly, constraint (46) handles the possible arrival of each train at its destination within its time window. Constraints (47) are flow conservation constraints for the nodes in *StatTimeNodes*, while constraint (48) restricts arc capacities. Constraint set (49) is the track capacity constraints. These constraints ensure that there is not more than one train traveling on a track segment at any time. This is accomplished by ensuring that at most one of

the arcs passing through the same time instant at a particular track segment carries a unit flow (Şahin et al. [45]). Note that in the generalized formulation, the flow variable on an arc is the sum of multicommodity flow variables on that arc, i.e., $x_{ij} = \sum_t x_{ij}^t$. The first term in constraint (49) represents the sum of flows on the arcs in the outbound direction (from station i to station $i+1$) during time period k. The second term represents the flows on the arcs in the inbound direction (from station $i+1$ to station i) during time period k. In total, the number of such constraints is equal to the number of discrete time periods (i.e., $(|S|-1)*|Q|$).

We now briefly discuss the size of the problem for a typical railroad application. The number of stations and sidings $|S|$ in the network can reach 100. Proper time representation to accurately represent traffic may require time to be discretized in periods of 10 minutes or less. For a planning horizon of two days (48 hours), the resulting network has about 28,800 nodes. Fifty trains traveling along the network imply some 720,000 or more binary decision variables. Thus, even a moderate-size problem turns out to be very large to be solved to optimality using existing commercial-level IP software.

6.2. Solution Approach

Motivated by developing an algorithm to help dispatchers make decisions in real-time scheduling, which requires that effective solutions be obtained quickly (usually in no more than 5–10 minutes), Şahin et al. [45] proposed an IP-based heuristic to solve the train dispatching problem. This heuristic is based on observation of the performance of CPLEX version 8.1 to solve different instances of the train dispatching problem. It is observed that for moderately small instances, which are measured in terms of the number of sidings, the length of the time horizon, the number of trains, and the number of potential conflicts, CPLEX performs well, reaching the optimal solution in a few minutes. An important aspect of the formulation is that larger maximum delay allowances lead to larger problems due to the increase of the number of arcs in the network, representing potential alternative solutions to the conflicts. Thus, decreasing maximum delay allowances tightens the formulation without affecting the feasibility. As the size of the problem increases and/or as longer maximum train delays are allowed, CPLEX fails to quickly reach the optimal solution, though a feasible solution can be found in a reasonably short amount of time, usually in less than a minute.

It should be noted that real-time train traffic control is almost overwhelmingly dynamic; unexpected and unpredictable events occur frequently, bringing perturbations that cause trains to be delayed all the time. This need for real-time optimization suggests that a major effort to determine a global optimal solution for all trains over the whole scheduling horizon (typically ranging from 8–12 hours to up to 1–2 days) may not be worth the effort. These unpredictable events may make it quite unlikely for some trains to strictly follow their original schedules, particularly in the long term. On the other hand, not taking into consideration a sufficiently long scheduling horizon may lead to optimal or near-optimal decisions to resolve conflicts being made based solely in the short-term range. Additionally, these decisions made with only the short term in mind may even turn infeasible in the long term, not to mention that in real-time dispatching it may be undesirable and inconvenient to change the entire train schedule by reoptimizing the whole problem every time a train is delayed. All these issues led to the development of a heuristic that is based on the IP formulation, which relies on available, off-the-shelf optimization software to find effective solutions.

The basic idea in this method consists of gradually reducing the maximum total allowable delays (md^t) for all trains based on new feasible integer solutions that are progressively obtained by CPLEX. Initially, we divide the scheduling horizon q into two periods, the short and the long terms. Let q_s be the length of the short term ($q_s \leq q$); consequently $q_l = (q - q_s)$ defines the remaining length of the scheduling horizon (i.e., the long term). We define δ_s

FIGURE 14. The IP-based heuristic for the train dispatching problem.

algorithm *IP-Based Heuristic for TDP*;
begin
find the first initial feasible solution for the IP formulation using CPLEX;
while the current solution is not locally optimal **or** CPU time limit is not reached **do**
for each train t
set $md^t = \min(d^t + \delta_1, md^t)$ for the short term (periods 1- q_s);
set $md^t = \min(d^t + \delta_2, md^t)$ for the long term (periods q_s+1, q);
end; {for each train t}
process CPLEX for σ seconds and find the current best feasible solution;
end;
end;

and δ_l as increments to the train delays obtained in a feasible solution for the short and long terms, respectively. Suppose we find a feasible solution in which a train t is delayed d^t units of time ($d^t \leq md^t$). The idea is to make the mathematical formulation progressively tighter, to make it converge faster to a good solution. In this case, the maximum allowed delay for train t is reduced from md^t to the minimum of $(d^t + \delta_s)$ and md^t for the time periods within the short term. Similarly, it is reduced from md^t to the minimum of $(d^t + \delta_l)$ and md^t in the long term where $\delta_l \leq \delta_s$. Note that the total delay along the whole scheduling horizon is still less than the maximum delay allowed for the short term. Therefore, a feasible solution is always guaranteed, and the problem is tighter. The optimization process is then restarted for this tighter problem and proceeds for a fixed time period σ. With this method, the chances that a better feasible solution is found in a shorter amount of time are increased. The same process is repeated until an optimal solution for the progressively tightened problem is found or a maximum total CPU time is reached. Figure 14 formalizes the IP-based heuristic for the train dispatching problem.

6.3. Computational Results

The paper Şahin et al. [45] reports extensive computational results for a set of generated test problems. These tests also consider two other heuristics proposed by the authors: a simulation-based construction heuristic and a greedy enumeration heuristic. The results show that the solution quality of the IP-based heuristic is very satisfactory and the solution times are competitive even when compared to the other simpler heuristics. In this section, we only discuss the results of real-life problem of a major Brazilian railroad transporting heavy-haul freight (iron ore unit trains with low-interval, frequent trains). The data is obtained from Leal et al. [36]. The problem comprises a single-line track that links two major stations with 23 intermediate stations or sidings where trains can be pulled over. 25 scheduled trains are to be dispatched over a 24-hour period. The trains run in both directions. The train routes, the travel times in each section, and the scheduled departure times are all known. Our objective is to minimize the total unweighted delay of all the trains. We set the maximum allowed delay equal to four hours for all the trains traveling between the two endpoints; for the remaining trains traveling between a pair of intermediate stations, we set the maximum allowed delay as proportional to this value based on the total travel time. Times are discretized in periods of only five minutes to produce a precise timetable for the trains.

Table 6 presents the results obtained for this real-world problem. The results show that the IP-based heuristic achieved the best result (0.06% above the optimal, corresponding to only five minutes of additional delay), significantly improving those obtained manually by the dispatchers, as well as the previous results of Leal et al. [36].

Table 6. Results for a real-world problem.

	Total travel time (minutes)	Total delay (minutes)	Deviation from optimal (%)	CPU time (seconds)
Dispatchers solution	14,119	6,311	71.35	—
Optimal IP solution	8,240	432	—	820
IP-based heuristic	8,245	437	0.06	173
Heuristic #1*	8,766	958	6.38	Not available
Heuristic #2*	8,755	947	6.25	Not available

*Obtained from Leal et al. [36].

6.4. Concluding Remarks

The novelty of the proposed IP formulation is based on the fact that it can easily handle several practical constraints without making the formulation more complex. The structure of the formulation does not change by adding these new constraints, because the variables are based on a space-time network representation rather than a conventional transformation of the variables. The complex railroad network infrastructures, such as double-lines and intersecting railroads, are easily handled with network representation, which avoids the complications in the formulation.

These approaches should be evaluated from two different perspectives: a master planning tool and a real-time decision-making tool. Our experiments indicate that with enough technology and advanced computational infrastructure that can support the most recent solver technology, the IP formulation provides satisfactory results as a master planning tool. The IP-based heuristic should also be considered a trustworthy master planning tool because the experiments confirm that it provides high-quality solutions with significantly less computational effort.

A real-time decision tool is a critical requirement for this problem because of unexpected delays and dynamically changing parameters that require train schedules to be constantly updated. Even the perfect master planning tools have no value unless they can support real-time decisions. The flexibility to change the parameters and the dynamic feature of the proposed approach make it a quick aid tool that the dispatcher can use to evaluate the effectiveness of real-time decisions.

7. Crew Scheduling Problem

Crew scheduling is a well known and widely studied problem in operations research, especially related to airlines and public transit systems. Railway crew scheduling problems have recently started to appear in operations research literature. The nature of these problems is highly dependent on the environmental factors (such as country, passenger/freight railroad, labor laws, and specific union agreements), and the majority of the studies conducted naturally focus on the specific characteristics related to the particular problem being studied, thus making it more difficult to apply it to a broad class of problems. In this section, we discuss the railroad crew scheduling problem according to the North American system, which is mainly composed of freight carrier operations.

Railroads employ thousands of units of manpower for the efficient transportation of the shipments (e.g., loading, unloading, yards, trains, traffic control, etc.). Hence, for a railroad, the manpower cost is a significant component of the overall transportation cost. A significant part of these employees, usually highly skilled, are involved in operating the trains on their planned timetable. Generally an engineer (who operates the locomotives) and a conductor (who coordinates the movement) are deployed in a train in its trip from its origin to its destination; this team is called a *crew/turn*. The *crew scheduling* (also called *crew planning* or *crew balancing*) problem is to schedule the crews in such a way that trains can be operated

at their schedule at minimum crew cost. In rail industry, the crew scheduling problem arises at two different levels:

(i) strategic level, in which decisions are made with respect to the number of crews required to operate the complete schedule of a long planning horizon, and

(ii) tactical level, in which for a given set of crews, assignments to trains are decided on a shorter time horizon (generally weekly).

In this section, we only focus on the tactical-level problem. Thus, the crew availability is given, and the aim is to determine crew assignment to operate trains in the schedule of a fairly short period of time (a week or shorter).

The two most common approaches in formulating and solving crew scheduling problems are based on column generation and set covering. A review of network models for crew scheduling in mass transportation systems can be found in Carraresi and Gallo [15]. In two recent studies on railroad crew scheduling problems, Chu and Chan [17] propose a network flow based algorithm for the Hong Kong Light Rail System and Caprara et al. [13] study a real-life case of the Italian railroad.

7.1. Characteristics of the Railroad Crew System

Similar to other applications of crew scheduling, railway crew scheduling is a complex problem. Not only the issues related with the rail traffic restrictions and/or requirements, but also contractual agreements and abiding laws should be taken into account in a feasible schedule. One particular characteristic of the North American railroads crew management system is the division of the rail network into *crew districts* that constitute a subset of terminals (nodes) and rail links (arcs). The crews are also grouped into *crew pools* or *crew types*. A crew type is usually characterized by a home terminal and a particular set of trains to operate. Every crew type has a terminal called the home terminal. The railroad does not incur any lodging cost when a crew member is at this terminal. All terminals other than the crew's home terminal are called *away terminals*. The rest of a crew member at any of these away terminals requires the railroad to take care of the necessary lodging arrangements, generally a hotel. In addition, further crew-type classification might be also based on a set of contractual rules or labor laws. For instance, two different crew types with the same home terminal who can operate the trains with same origin and destination are differentiated by their duty time period. While one crew type can be scheduled to operate the trains departing only before 6 pm, the other can be scheduled to work irrespective of the time.

On-duty time corresponds to the time at which a crew reports to be on duty. It does not necessarily correspond to the departure time of the train, because there may be some tasks to be performed by the crew members before the train departs. Similar to the tasks performed before the departure of the train, there are some tasks that are required to be performed upon the arrival of the train at the destination. Therefore, a crew will have to work after the arrival of the train. The time by which a crew finishes these tasks is called *tie-up time*. Accordingly, *duty period* of the crew starts at on-duty time and ends at tie-up time. The length of this period is also referred to as the journey length of the train.

Crew deadheading and *crew detention* are two special cases when the railroad is to pay the hourly rate to a crew member even while the crew is actually idle (i.e., not assigned to duty). Normally, a crew gets on to a train at the home terminal and takes it to the away terminal. After taking adequate rest at the away terminal, it is assigned to another train that is scheduled to another away terminal or back to the home terminal. However, either to meet the shortage of crew at a terminal or to improve the cost efficiency, a crew may be sent from one terminal to another by some other mode of transportation (e.g., via a taxi). All such idle travels of crews are called crew deadheadings. In each deadheading, the railroad incurs the cost of crew time traveling idle as well as the extra expenditure related to the movement of the crew (e.g., renting a taxi). Whenever a crew is released from duty at an away terminal, they stay in a hotel for an adequate rest period before being assigned to the

next train. After this minimum required rest period, there is a prespecified period in which crew can be called to duty at any time. However, if a crew will have to stay at an away terminal after this period, they are entitled to salary and, hence, the railroad has to incur extra cost for the rest of the crew. The period during which crew is paid salary while taking rest or idling at the away terminal is called *crew detention*. For example, let us assume that in a crew district, the minimum rest period at an away terminal is eight hours and the call period is eight hours. If a crew is not working for 18 hours at an away terminal before being assigned to a train, then the crew detention is 2 hours.

7.1.1. Contractual Restrictions. The problem of crew scheduling is apparently involved with workers. Hence, the work environment is bounded by the regulations imposed by the contractual agreements, labor regulations, and union agreements. Though such restrictions may vary slightly from one district to another or from one crew type to another, the regulations impose similar constraints to the work environment. Below, we list a set of restrictions that are commonly applicable to all crew districts and types:

(i) Duty period of a crew cannot exceed 12 hours. For an assignment to a train, the duty period includes the time elapsed between on-duty time and tie-up time, as mentioned before.

(ii) Whenever a crew member is released from duty at the home terminal or has been deadheaded to the home terminal, he/she can resume another duty only after 12 hours (10 hours rest followed by 2 hours call period) if the duty period is greater than 10 hours, and 10 hours (8 hours rest followed by 2 hours call period) if the duty period is less than or equal to 10 hours.

(iii) Irrespective of the duration of the stay of a crew at the home terminal, they are not paid any salary for the rest period.

(iv) Whenever a crew member is released from duty at the away terminal, he/she must go for a minimum of 8 hours of rest unless either of the following exceptions occurs. The first exception arises if the total time period corresponding to the recent travel time from the home terminal followed by a rest time less than 4 hours, plus travel time of a new assignment back home, is shorter than 12 hours. The second exception arises if the total time corresponding to the recent travel time from the home terminal plus travel time of a new assignment back home is less than 12 hours, and the rest time in-between the assignments is more than 4 hours. In both cases, there should be no other crew at the away terminal who is completely rested.

(iv) The deployment of crew must follow the first-in-first-out (FIFO) rule at both the home terminal and the away terminals. This rule implies that a crew with the maximum rest period among all the crews that are taking rest at the on-duty time of the next train should be assigned to that train.

The European railroad system runs strictly on schedule, mostly due to the fact that the schedule is composed of a mixture of freight and passenger trains, and the travel times are significantly shorter than those of the North American railroads. However, it is likely and probable to experience delays in the North American system for several reasons. Although this is highly unwanted, the system still should be capable of handling such situations by incurring appropriate costs. This is also an important factor from a quality-of-service perspective. One reason for departure delays is the unavailability of crew to run the train on schedule, which we refer to here as *crew delay*.

7.2. Mathematical Formulation

The railroad crew scheduling problem is to assign available crews to the trains over the planning horizon (generally weekly), such that the total costs of crew, deadheading, detention and crew delay are minimum while honoring the train schedule, the rail traffic constraints,

and contractual requirements. We first briefly describe the input, objective function, constraints, and decision variables for the railroad crew scheduling problem. We then develop a network representation of the problem and formulate a network flow model for this network. For simplicity, we formulate the problem for a single crew type of one district only. The home terminal of this crew type is either the origin or the destination of the trains considered in the schedule. All other terminals in the schedule are considered away terminals for this crew type. The network representation of the problem and the formulation can easily be extended to more crew types. Now, we briefly discuss the input requirements, decision variables, objective function, and constraints of the problem.

Input requirements for crew scheduling can be listed as follows:

Train schedule: This includes information about each train: its departure time at its origin and its arrival time at its destination. We also require the on-duty time and tie-up time for each train, and also the information.

Crew availability: This includes the name and type of each crew, its initial position (the terminal and time at which it finished its previous duty) at the beginning of the planning period, and the hours of duty done before.

Cost parameters: This includes per-hour salary of crew, per-hour taxi cost (deadhead), per-hour detention charge, and per-hour crew delay cost.

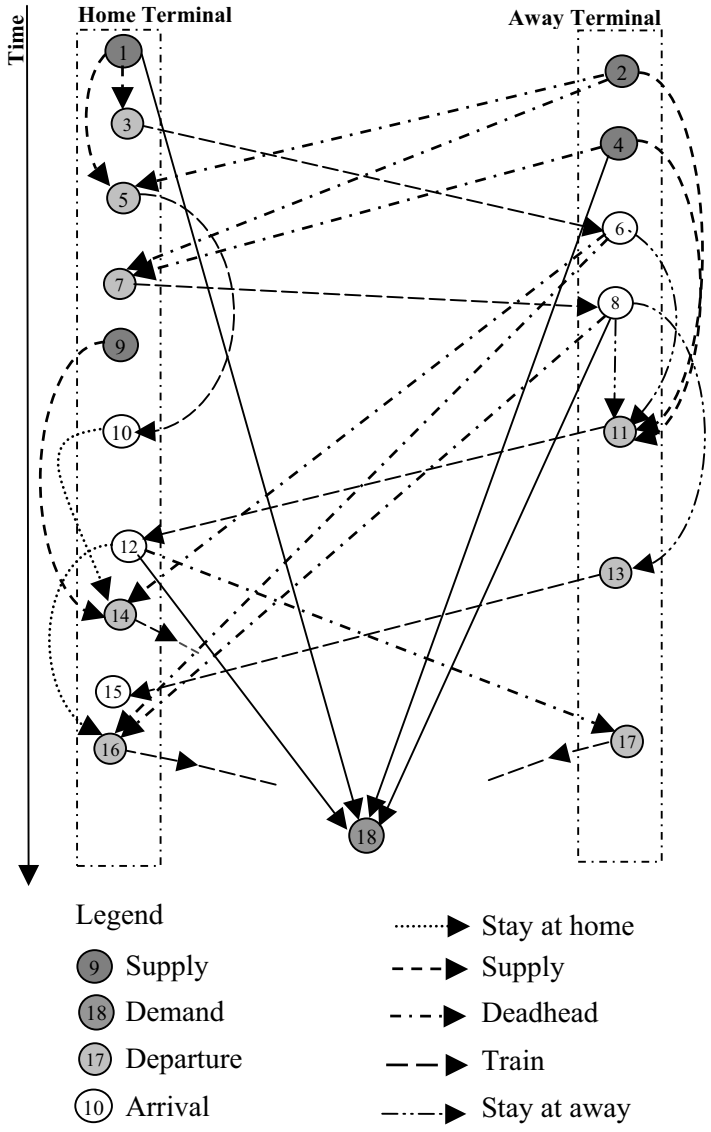
In the crew scheduling problem, we make decisions regarding the assignment of crews to trains, deadheading of crews, and crew delay of trains. The constraints can be considered in two groups: operational constraints and contractual requirements. The operational constraints ensure that every train gets a qualified crew to operate it, while a crew is not assigned to more than one train at the same time under the setting imposed by the crew districts and types. Therefore, only crews of certain crew types can be assigned to a certain train. Assignment of crews to trains must, in addition, satisfy the contractual requirements. The operational constraints of the model are handled by the network flow formulation, while the contractual restrictions are honored in the network construction phase. The objective of the crew scheduling problem is to minimize the total cost of salary to the crew, the cost of deadheading, the cost of crew detentions, and the cost of crew delay.

In this section, we discuss the network flow-based approach for the crew scheduling problem in Vaidyanathan et al. [48]. The problem is first represented by a space-time network in which (i) the nodes represent train origins and destinations and crew availability, (ii) the arcs are built based on possible and feasible assignments of crew to trains, and (iii) the cost parameters of the objective function are reflected in the network by the arc costs. Vaidyanathan et al. [48] then formulate the problem as a minimum-cost network flow problem in which each crew is represented by a commodity that flows.

7.2.1. Constructing the Space-Time Network. In the space-time network, each node has a time attribute attached to it in addition to its location attribute. We demonstrate the network representation with an example in Figure 15. To start with, the complete planning horizon is discretized in appropriately small time units. Then, the arrival and departure times of trains are mapped onto this discrete-time planning horizon.

Supply nodes represent the crew availability. Each crew member is denoted by a supply node whose time attribute is the time at which the crew is available, and whose location attribute corresponds to the terminal from where the crew can be released for duty. Because each supply node represents a crew member, the supply of the node is set to 1. We need only one demand node with a time attribute equal to the end of the planning horizon, and no specific location attribute. The demand of this node is set equal to the number of available crew members (i.e., the number of supply nodes). Departure nodes and arrival nodes correspond to the departures and arrivals of trains, respectively. The time attributes are attached by considering on-duty time (tie up time) for departure (arrival) nodes while the location attributes simply represent the origin (destination) terminals.

FIGURE 15. Crew scheduling network for a single district and a single crew type.



In Figure 15, arcs from supply nodes to departure nodes represent the possible first duty assignment of a crew to a train. Train arcs are constructed for each train from its departure node to its arrival node, and the arc costs are calculated based on the hourly salary and the travel time of the train. Deadhead arcs can be either from a supply node or arrival node to a train departure node representing the deadhead movement of a crew member with the associated deadheading cost. Stay arcs at the home (away) terminal are constructed from an arrival node to a departure node at the same terminal. While constructing the stay arcs, all possibilities of a crew member being assigned from an arriving train to a departing train should be considered, and the arc costs are calculated based on the length of the stay. Both deadhead arcs and stay arcs should comply with the contractual restrictions, hence, the network construction honors the associated constraints by creating only allowable arcs. Finally, the demand arcs are constructed from any of the arrival nodes and supply nodes to the single demand node by taking into account all possible last assignments of the crew members in the planning horizon.

FIFO policy cannot be easily handled while constructing the network. Figure 15 shows an example network where all rules except FIFO policy are considered as explained above. However, we will able to resolve this issue by adding a side constraint to the minimum-cost network flow formulation.

7.2.2. Integer Programming Formulation. The constructed network is used to formulate a minimum-cost network flow problem in which each crew corresponds to a commodity of the network. The formulation honors the operational constraints of the problem with flow balance and arc capacity constraints. In the following formulation, N is the node set where $i \in N$ represents a node and A is the arc set where $a \in A$ represents an arc; $I(i)$ $(O(i))$ is again the set of incoming (outgoing) arcs to (from) node i, and n is the total number of crew members available for the planning horizon. Let decision variable x_a represent the flow on arc a, and c_a be the associated arc cost. To take care of FIFO constraints, we define A_a as the set of arcs on which there cannot be positive flow if there is a flow on arc a.

$$\text{Minimize} \quad \sum_{a \in A} c_a x_a \tag{51}$$

$$\text{subject to} \quad \sum_{a \in O(i)} x_a = 1 \quad \forall i \in N \text{ and } i \text{ is a departure node} \tag{52}$$

$$\sum_{a \in O(i)} x_a = 1 \quad \forall i \in N \text{ and } i \text{ is a supply node} \tag{53}$$

$$\sum_{a \in I(i)} x_a = n \quad \forall i \in N \text{ and } i \text{ is the demand node} \tag{54}$$

$$\sum_{a \in I(i)} x_a = \sum_{a \in O(i)} x_a \quad \forall i \in N \text{ and } i \text{ is a departure or arrival node} \tag{55}$$

$$\sum_{a' \in A_a} x'_a \le M(1 - x_a) \quad \forall a \in A \tag{56}$$

$$x_a \in \{0, 1\} \quad \forall a \in A. \tag{57}$$

The objective function (51) minimizes the total arc costs, hence the total crew salary, deadheading, and detentions. Constraint (52) ensures that each train gets a crew member. Contraints (53) and (54) satisfy the supply and demand balance of the supply and demand nodes, respectively. Constraint (55) satisfies the flow balance for all remaining nodes. Constraint (56) represents the FIFO policy for the crew assignments based on the prohibited arc flows for each arc. Constraint (57) is the integrality constraint for the decision variable.

It should be noted that so far we have not considered the crew delay in our space-time network, and hence in the formulation. To induce the crew delay possibility, we also create those arcs that may violate the minimum-stay requirement, assuming that if there is any flow on these arcs, it will correspond to the delay of a train at its head node, such that the minimum-stay requirement is met. In addition to other costs, we put a penalty equivalent to the crew delay cost on these arcs.

7.3. Solution Method and Computational Results

Railroads are obviously more interested in having tools that can be used in real time and that can produce a reasonably good solution. The formulation without the FIFO constraints is a simple minimum-cost network flow formulation. However, FIFO constraints bring a challenge, especially in terms of the solution time. The experiments in Vaidyanathan et al. [48] show that the problem without the FIFO constraints could be solved within five seconds to optimality, using CPLEX 8.0, for real-life problem data consisting of 436 trains, 99 crews,

Table 7. Results for a real-life data of the crew scheduling problem.

Criteria	Current solution	UF solution	Improvement (%)
Crew delay	48 hours	0 hours	100
Deadheadings	322 hours	200 hours	37.9
Crew detention	88 hours	53 hours	39.8

and three operating terminals over a period of seven days. Hence, they propose a postprocessing procedure that repairs the solution of the problem without the FIFO constraints, and finds an approximate solution to the problem with FIFO constraints.

The postprocessing procedure starts by listing the crew assignments according to the arrival times of incoming trains and departure times of outgoing trains at all terminals. Starting from the beginning of the planning horizon, we search through the end of the list by detecting the assignments that violate FIFO constraints. For each pair of assignments that violates the constraints, we basically first swap the crew assignment for the pair and then update the future path of both crews accordingly by exchanging the future assignments between the two crews. Note that during the postprocessing the deadheading between two terminals is considered as a train between those two terminals and handled in the same way as the other trains are. In addition, both the crew delay and number of deadheads remain unchanged.

This algorithm is tested on the real-life data we mentioned previously. We compare the results in Vaidyanathan et al. [48] with the current solution used by the railroad company on three criteria: crew delay, deadheadings, and crew detention. As indicated by the values in Table 7, the solution in Vaidyanathan et al. [48] (UF Solution) outperforms the current solution of the railroad on all criteria.

7.4. Concluding Remarks

For the sake of simplicity and clarity, the formulation of the crew scheduling problem given in this section is for a single crew type and a single crew district. Indeed, the crew scheduling problem of a particular railroad is involved with multiple crew types of several crew districts. Some crew districts share a set of common trains, and different crew types of the same district can operate the same set of trains unless prohibited by the rules that particularly govern the crew type. Hence, the planning problem should be capable of handling multiple crew types and multiple crew districts. The network flow formulation we discuss here can easily be adapted for this case. The resulting network is a multicommodity flow network, where each crew type corresponds to a commodity type. Hence, the network flow formulation is a multicommodity network flow problem. Note that elimination of FIFO constraints and the associated postprocessing procedure is still valid to attain a feasible solution to the problem. The challenge here lies in finding an efficient solution method for the multicommodity network flow problem (without FIFO constraints), which is known to be NP-hard. However, the problems and formulations we discuss in other sections also face this challenge, and some efficient algorithms have overcome this difficulty by designing particular algorithms for each class of problems.

8. Conclusions

In this chapter, we deal with the six most important planning and scheduling problems for railroads, and describe the main achievements that have been obtained so far. We initially give a detailed characterization of each problem, its context and importance, followed by the mathematical formulation, which is based on a convenient network flow representation; in addition, we discuss the main issues involved that make real instances difficult to model and solve. We also present cutting-edge algorithms that effectively solve instances found

in practice by railroads, as well as the results obtained, thus allowing railroads to achieve substantial savings.

We hope to have successfully evidenced to the reader that railroads pose some of the most exciting and challenging opportunities for the application of operations research tools and methodologies. In addition, our aim is to have demonstrated that difficult, combinatorial, NP-hard problems that appear in practice can be effectively solved by the proper choice of efficient and effective techniques, cleverly designed to take advantage of the special characteristics and nature of each problem.

We also want to emphasize here that if operations research is to be acknowledged as *the science of better* and is effectively applied to solve real-world problems and to deliver premium value to the world's organizations, we must go beyond the current paradigms that drive the problems we choose to tackle, the way we model them, and how they are solved.

To summarize, if we want to develop and provide operations research tools that help organizations find near-optimal solutions that can be applied and used in practice, resulting in substantial savings, not to mention the benefits in terms of quality, improved level of service, and customer satisfaction and reliability, to name a few, we must be *innovative* in modeling and solving them. There are plenty of challenging opportunities involving the overall railroad business, given its intrinsic complexity and the interrelationships between problems, as we tried to demonstrate in this chapter.

References

[1] B. Adenso-Diaz, M. O. Gonzalez, and P. Gonzalez-Torre. On-line timetable re-scheduling in regional train services. *Transportation Research B* 33:387–398, 1999.

[2] R. K. Ahuja, K. C. Jha, and J. Liu. Solving real-life railroad blocking problems. *Operations Research*, 2004. Forthcoming.

[3] R. K. Ahuja, J. Liu, and G. Sahin. Railroads yard location problem. Working paper, Innovative Scheduling Systems, Inc., Gainesville, FL, 2005.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, 1993.

[5] R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics* 123:75–102, 2002.

[6] R. K. Ahuja, J. Liu, A. Mukherjee, J. B. Orlin, D. Sharma, and L. A. Shughart. Solving real-life locomotive scheduling problems. *Transportation Science*. Forthcoming.

[7] A. Assad. Models for rail transportation. *Transportation Research A* 14:205–220, 1980.

[8] C. Barnhart, H. Jin, and P. H. Vance. Railroad blocking: A network design application. *Operations Research* 48:603–614, 2000.

[9] L. D. Bodin, B. L. Golden, A. D. Schuster, and W. Romig. A model for the blocking of trains. *Transportation Research B* 14:115–120, 1980.

[10] U. Brännlund, P. O. Lindberg, A. Nõu, and J.-E. Nilsson. Railway timetabling using Lagrangian relaxation. *Transportation Science* 32(4):358–369, 1998.

[11] X. Cai, C. J. Goh, and A. I. Mees. Greedy heuristics for rapid scheduling of trains on a single track. *IIE Transactions* 30:481–493, 1998.

[12] A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research* 50(5):851–861, 2002.

[13] A. Caprara, M. Fischetti, P. L. Guida, P. Toth, and D. Vigo. Solution of large-scale railway crew planning problems: The Italian experience. *Lecture Notes in Economics and Mathematical Systems*, Vol. 471, 1–18, 1999.

[14] M. Carey and D. Lockwood. A model, algorithms and strategy for train pathing. *Journal of the Operational Research Society* 46:988–1005, 1995.

[15] P. Carraresi and G. Gallo. Network models for vehicle and crew scheduling. *European Journal of Operational Research* 16:139–151, 1984.

[16] K. C. Chih, M. A. Hornung, M. S. Rothenberg, and A. L. Kornhauser. Implementation of a real time locomotive distribution system. T. K. S. Murthy, R. E. Rivier, G. F. List, and J. Mikolaj, eds. *Computer Applications in Railway Planning and Management*. Computational Mechanics Publications, Southampton, UK, 1990.

[17] S. C. K. Chu and E. C. H. Chan. Crew scheduling of light rail transit in Hong Kong: From modeling to implementation. *Computers and Operations Research* 25(11):887–894, 1998.

[18] J.-F. Cordeau, F. Soumis, and J. Desrosiers. A Benders' decomposition approach for the locomotive and car assignment problem. Technical Report G-98-35, GERAD, École de Hautes Études Commerciales de Montréal, Montréal, Quebec, Canada, 1998.

[19] J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science* 32:988–1005, 1998.

[20] T. G. Crainic and J.-M. Rosseau. Multimode, multicommodity freight transportation: A general modelling and algorithmic framework for the service network design problem. *Transportation Research B* 20(3):225–242, 1986.

[21] T. G. Crainic, J. A. Ferland, J.-M. Rousseau. A tactical planning model for rail freight transportation. *Transportation Science* 18(2):165–184, 1984.

[22] M. Fischetti and P. Toth. A package for locomotive scheduling, Technical Report DEIS-OR-97-16, University of Bologna, Bologna, Italy, 1997.

[23] M. Florian, G. Bushell, J. Ferland, G. Guerin, and L. Nastansky. The engine scheduling problem in a railway network. *INFOR* 14:121–138, 1976.

[24] M. A. Forbes, J. N. Holt, and A. M. Watts. Exact solution of locomotive scheduling problems. *Journal of Operational Research Society* 42:825–831, 1991.

[25] M. F. Gorman. An application of genetic and tabu searches to the freight railroad operating plan problem. *Annals of Operations Research* 78:51–69, 1998.

[26] E. Haghani. Rail freight transportation: A review of recent optimization models for train routing and empty car distribution. *Journal of Advanced Transportation* 21:147–172, 1987.

[27] A. E. Haghani. Formulation and solution of a combined train routing and makeup, and empty car distribution model. *Transportation Research B* 23(6):433–452, 1989.

[28] A. Higgins, E. Kozan, and L. Ferreira. Optimal scheduling of trains on a single line track. *Transportation Research B* 30:147–161, 1996.

[29] A. Higgins, E. Kozan, and L. Ferreira. Heuristic techniques for single line tran scheduling. *Journal of Heuristics* 3:43–62, 1997.

[30] P. Ireland, R. Case, J. Fallis, C. Van Dyke, J. Kuehn, and M. Meketon. The Canadian Pacific Railway transforms operations research by using models to develop its operating plans. *Interfaces* 34(1):5–14, 2004.

[31] M. H. Keaton. Designing optimal railroad operating plans—Lagrangian-relaxation and heuristic approaches. *Transportation Research B* 23(6):415–431, 1989.

[32] M. H. Keaton. Service-cost tradeoffs for carload freight traffic in the United States rail industry. *Transportation Research A* 25(6):363–374, 1991.

[33] M. H. Keaton. Designing railroad operating plans—A dual adjustment method for implementing lagrangian-relaxation. *Transportation Science* 26(4):263–279, 1992.

[34] D. Kraay and P. Harker. Real-time scheduling of freight networks. *Transportation Research B* 29(3):213–229, 1994.

[35] L. G. Kroon and L.W. Peeters. A variable trip time model for cyclic railway timetabling. *Transportation Science* 37(2):198–212, 2003.

[36] J. E. Leal, A. C. Soares, and L. C. N. Nunes. A heuristic approach for the train scheduling problem on single railway tracks. *ANPET Congress*, Vol. 2. ANPET, Brazil, 945–954, 2004.

[37] M. Newman, L. Nozick, and C. A. Yano. Optimization in the rail industry. P. P. Pardalos and M. G. C. Resende, eds. *Handbook of Applied Optimization*. Oxford University Press, New York, 2002.

[38] H. N. Newton. Network design under budget constraints with application to the railroad blocking problem. Ph.D. thesis, Industrial and Systems Engineering Department, Auburn University, Auburn, AL, 1996.

[39] H. N. Newton, C. Barnhart, and P. M. Vance. Constructing railroad blocking plans to minimize handling costs. *Transportation Science* 32:330–345, 1998.

[40] A. Nou, J. Desrosiers, and F. Soumis. Weekly locomotive scheduling at Swedish State Railways. Technical Report G-97-35, GERAD, École des Hautes Études Commerciales de Montréal, Montréal, Quebec, Canada, 1997.

[41] K. V. Ramani. An information system for allocating coach stock on Indian Railways. *Interfaces* 11(3):44–51, 1981.

[42] R. E. Rivier and P. Tzieropoulos. Interactive Graphic Models for Railway Operational Planning. M. Florian, ed. *The Practice of Transportation Planning*. Elsevier Science Publishers, Amsterdam, 1984.

[43] R. E. Rivier and P. Tzieropoulos. Computer-aided planning of railway networks, lines and stations. T. K. S. Murthy, L. S. Lawrence, and R. E. Rivier, eds. *Computers in Railway Management*. Computational Mechanics Publications, Berlin, Germany, 1987.

[44] İ. Şahin. Railway traffic control and train scheduling based on inter-train conflict management. *Transportation Research B* 33:511–534, 1999.

[45] G. Şahin, R. K. Ahuja, and C. B. Cunha. New approaches for the train dispatching problem. Working paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, 2004.

[46] R. L. Sauder and W. Westerman. Computer aided train dispatching: Decision support through optimization. *Interfaces* 13(6):24–37, 1993.

[47] S. Smith and Y. Sheffi. Locomotive scheduling under uncertain demand. *Transportation Research Record* 1251:45–53, 1988.

[48] B. Vaidyanathan, R. K. Ahuja, and K. C. Jha. A network flow approach for railroad crew scheduling. Working paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, 2004.

[49] M. B. Wright. Applying stochastic algorithms to a locomotive scheduling problem. *Journal of Operational Research Society* 40:187–192, 1989.

[50] G. Yu. *Operations Research in the Airline Industry*. Kluwer Academic Publishers, Boston, MA, 1998.

[51] K. Ziarati, F. Soumis, J. Desrosiers, and M. M. Solomon. A branch-first, cut-second approach for locomotive assignment. *Management Science* 45:1156–1168, 1999.

[52] K. Ziarati, F. Soumis, J. Desrosiers, S. Gelinas, and A. Saintonge. Locomotive assignment with heterogeneous consists at CN North America. *European Journal of Operational Research* 97:281–292, 1997.