TRANSPORTATION SCIENCE
Vol. 39, No. 4, November 2005, pp. 503–517
ISSN 0041-1655 | EISSN 1526-5447 | 05 | 3904 | 0503

DOI 10.1287/trsc.1050.0115
© 2005 INFORMS

Solving Real-Life Locomotive-Scheduling Problems

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

Jian Liu
Innovative Scheduling, GTEC, 2153 SE Hawthorne Road, Gainesville, Florida 32641,
liujian@innovativescheduling.com

James B. Orlin
Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139,
jorlin@mit.edu

Dushyant Sharma
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109,
dushyant@umich.edu

Larry A. Shughart
Innovative Scheduling, GTEC, 2153 SE Hawthorne Road, Gainesville, Florida 32641,
larry@innovativescheduling.com

In the locomotive-scheduling problem (or the locomotive-assignment problem), we must assign a *consist* (a set of locomotives) to each train in a preplanned train schedule so as to provide each train with sufficient locomotive power to pull the train from its origin to its destination. Locomotive-scheduling problems are among the most important problems in railroad scheduling. In this paper, we report the results of a study on the locomotive-scheduling problem as it is faced by CSX Transportation, a major U.S. railroad company. We consider the planning version of the locomotive-scheduling model (LSM) in which multiple types of locomotives exist, and we need to decide which set of locomotives should be assigned to each train. We present an integrated model that determines: the set of active and deadheaded locomotives for each train; the light-traveling locomotives from power sources to power sinks; and train-to-train connections (for which we specify which inbound trains and outbound trains can directly connect). An important feature of our model is that we explicitly consider *consist bustings* and *consistency*. A consist is said to be *busted* when a set of locomotives coming on an inbound train is broken into subsets to be reassigned to two or more outbound trains. A solution is *consistent* over a week if a train receives the same locomotive assignment each day that it runs. We will provide a mixed-integer programming (MIP) formulation of the locomotive-assignment problem. However, an MIP of this size cannot be solved to optimality or near optimality in acceptable running times using commercially available software. Using problem decomposition, integer programming, and very large-scale neighborhood search, we have developed a solution technique to solve this problem within 30 minutes of computation time on a Pentium III computer. Our solution obtained a potential savings of over 400 locomotives over the solution obtained by the in-house software developed by CSX.

Key words: rail transportation; integer programming; network optimization; train scheduling
History: Received: April 2002; revisions received: August 2003, September 2004; accepted: September 2004.

1. Introduction

Railroad transportation of goods plays an integral part in the U.S. economy, particularly in multimodal and container transportation. The rail transportation industry offers great potential for employing mathematical optimization techniques to solve the many problems that it currently faces. However, research in railroad scheduling has experienced a slow growth. Until recently, most contributions relied on simplified models or only applied to small instances, failing to incorporate the characteristics of real-life application. The intense competition that rail carriers face (most notably from trucking companies) and the ever-increasing speed of computers have motivated the 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. (See, for example, Cordeau, Toth, and Vigo 1998.) This paper concerns the development of new models and algorithms for solving real-life locomotive-scheduling problems faced by U.S. railroad companies.

For the locomotive-scheduling problem (or the locomotive-assignment problem), we must assign a *consist* (a set of locomotives) to each train in a preplanned train schedule so as to provide each train with sufficient power to pull the locomotives from its origin to its destination. Generally, a consist is chosen from a set of different fleet types (such as AC44, AC60, SD40, etc.). For example, a train's consist may be composed of two locomotives of types AC44 and one locomotive of type AC60.

Locomotive-scheduling problems are among the most important problems in railroad scheduling (Florian et al. 1976; Smith and Sheffi 1988; Chih et al. 1990; Forbes, Holt, and Watts 1991; Fischetti and Toth 1997; Nou, Desrosiers, and Soumis 1997; and Ziarati et al. 1997, 1999). Often, locomotive availability determines whether a train departs on time. With new locomotives costing in excess of $1.8 million, it is paramount to the railroad business that they be managed efficiently. The variety of locomotives and trains and the diverse geographic networks create very difficult combinatorial optimization problems. There are no satisfactory algorithms available to solve these problems. As a result, there are inefficiencies in locomotive management. However, through the development of better models and algorithms, substantial savings can be achieved. CSX Transportation has over 3,000 locomotives, which translates into a capital investment of over $5 billion, as well as over $1 billion in yearly maintenance and operational costs. Even a small percentage of savings in locomotive utilization could translate into substantial savings.

Locomotive-scheduling problems can be studied at the planning level or at the operational level. At the planning stage of the locomotive-scheduling problem, we assign locomotive types to various trains. The operational locomotive-scheduling model additionally takes into account the fueling and maintenance needs of the locomotives, which are ignored in the planning model. In this paper, we consider the planning version of the locomotive-scheduling model (LSM). We will now summarize the features of the LSM to provide the reader with a better understanding of the problem. A large railroad company has a train schedule that consists of several hundred trains with different weekly frequencies. Some trains run every day in a week, whereas others run less frequently. At CSX, there are several thousand train departures per week (assuming that we may count the same train running on different days multiple times). Many trains have long distances to travel and take several days to go from their origins to their destinations. To power these trains, CSX owns several thousand locomotives of different types with different horsepower and pulling capacities. In the LSM, we assign a set of locomotives to each train in the weekly train schedule so that each train receives sufficient tractive effort, or pulling power, and sufficient horsepower, or speed, and so that the assignment can be repeated indefinitely every week. At the same time, assigning a single 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.

An additional feature of the LSM is that some locomotives may *deadhead* on trains. Deadheaded locomotives do not pull the train; rather, they are pulled by active locomotives from one place to another. Deadheading plays an important role in locomotive-scheduling models because it allows extra locomotives to be moved from locations where they are in surplus to those where they are in short supply. Locomotives also *light travel*; that is, they travel on their own between different stations to reposition themselves between two successive assignments to trains. A set of locomotives in light travel forms a group, and one locomotive in the group pulls the others from an origin station to a destination station. Light travel is different from deadheading because it is not limited by the train schedule. In general, light travel is faster than deadheading. However, light travel is more costly, as a crew is required to operate the pulling locomotive, and the transportation does not generate any revenue because no cars are attached.

Because we assign a set of locomotives (or a *consist*) to trains, we need to account for *consist busting*. Whenever a train arrives at its destination, its consist 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, we say that there is a *train-to-train connection* between the inbound and outbound trains and no *consist busting* takes place. In the second case, consist busting takes place. Consist busting entails merging locomotives from inbound trains and regrouping them to make new consists. This is undesirable from several angles. First, consist busting requires additional locomotive and crew time to execute the moves. Second, consist busting 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, leading 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 bustings.

Another important feature of the locomotive-scheduling model is that we 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. If we make a train-to-train connection between two trains and if on three days in a week both trains run, then we want them to have the same train-to-train connection on all three days. Consistency of the locomotive assignment and train-to-train connections is highly desirable from an operational point of view. Consistency of the solution translates into rule-based dispatching and makes the job of a locomotive dispatcher relatively easy. For example, if a train is assigned three AC44 locomotives on each day it runs, the dispatcher can remember this rule and enforce it. If the train is assigned a different set of locomotives on different days, the assignments are harder to implement.

Our LSM model is substantially different from locomotive-scheduling models previously studied. Single-locomotive models have been studied by Forbes, Holt, and Watts (1991) and by Fischetti and Toth (1997). Multicommodity flow-based models for planning decisions have been studied by Florian et al. (1976), Smith and Sheffi (1988), and Nou, Desrosiers, and Soumis (1997). Multicommodity flow-based models for operational decisions have been developed by Chih et al. (1990) and by Ziarati et al. (1997, 1999). Our multicommodity flow-based model for planning decisions offers more features than any of the existing planning models.

The locomotive-scheduling problem is a very large-scale combinatorial optimization problem. We formulate it as a mixed-integer programming (MIP) problem, which is essentially an integer multicommodity flow problem with side constraints. We considered the locomotive-scheduling problem for CSX Transportation, which consisted of 3,324 trains originating from and terminating at 119 stations weekly and 3,316 locomotives of five locomotive types. Our formulation contains about 197,000 integer variables and 67,000 constraints, and is too large to be solved to optimality or near optimality using existing commercial-level MIP software. We therefore developed a heuristic methodology to solve the locomotive-scheduling problem. By using linear programming, mixed-integer programming, and very large-scale neighborhood search techniques, we have attempted to obtain very good solutions for the locomotive-scheduling problem. We solved the LSM in two stages. In the first stage, we modified the original problem so that all trains run seven days a week. This approximation of the original problem allows us to handle consistency constraints satisfactorily. In the second stage, we modified the solution of the first stage to solve the original problem so that trains do not run all seven days a week. We developed prototype software for our algorithms and compared our software with the in-house software used by CSX. In one representative instance for which CSX software required 1,614 locomotives to satisfy the train schedule, our software required only 1,210 locomotives, thereby achieving a savings of over 400 locomotives. We obtained similar improvements for other benchmark instances. In the solutions obtained by the CSX software, locomotives actively pull a train about 31.3% of the time, deadhead about 19.6% of the time, and idle at stations about 49.1% of the time. In the solutions obtained by our software, locomotives actively pull the train about 44.4% of the time, deadhead about 8.1% of the time, light travel about 0.8% of the time, and idle at stations about 46.7% of the time.

Though this research demonstrates that on the planning level significant reduction in the number of locomotives is possible, realizing these savings in practice requires additional work. Our research assumes that all trains run on time, and we assigned locomotive types to each train to minimize the systemwide costs. However, while assigning locomotives to trains in a real-time environment, we need to assign locomotive *tail numbers* (which are unique for each locomotive) to trains. For example, suppose that the planning model suggests that we assign two AC44s to train A. There may be six AC44 locomotives available for the assignment. Out of these six locomotives (with the same locomotive type but different tail numbers), we need to decide which two locomotives should be assigned to train A. To do so, we need to take into account the fueling requirement of locomotives. For example, a locomotive may have the fuel to go another 300 miles, but train A would not encounter any fueling station for the next 400 miles; clearly, we cannot assign this locomotive to train A. Another important issue is the maintenance of locomotives. Each locomotive needs to undergo periodic maintenance; regular maintenance is due every 3,000 miles and major maintenance is due every 10,000 miles. This maintenance can be done only at some locations, which are called *shops*. Thus, if a locomotive is due for regular 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. We therefore observe that a locomotive-assignment plan may have several corresponding locomotive schedules, some of which may satisfy fueling and maintenance requirements while others may not. Another factor that our original research ignored was train time uncertainty. We assumed that all trains run on time, whereas, in practice, trains are often late. How to account for delayed trains and how to update locomotive assignment with minimum-cost impact are central issues we need to resolve before our planning model can be applied. We are currently doing research to resolve these issues, and when successful, our planning model will prove itself quite valuable.

2. Problem Details

In this section, we will provide the details and notation of the locomotive-scheduling problem that we used for CSX.

Locomotive Data. A railroad company typically has several different types of locomotives with different pulling and cost characteristics and different numbers of axles (often ranging from four to nine). We denote by K the set of all the locomotive types and use the index k to represent a particular locomotive type. We associate the following data with each locomotive type $k \in K$: (i) h^k: the horsepower provided by a locomotive of type k; (ii) λ^k: the number of axles in a locomotive of type k; (iii) G^k: the weekly ownership cost for a locomotive of type k; and (iv) B^k: fleet size of locomotives of type k, that is, the number of locomotives available for assignment.

Train Data. Locomotives pull a set L of trains from their origins to their destinations. Trains have different weekly frequencies; some trains run every day, while others run less frequently. We will consider the same train running on different days as different trains; that is, if a train runs five days a week, we will consider it as five different trains. We use the index l to denote a specific train. We associate the following data for each train: (i) *dep-time*(l): the departure time for the train l; (ii) *arr-time*(l): the arrival time for train l (in the same format as the *dep-time*(l)); (iii) *dep-station*(l): the departure station for train l; (iv) *arr-station*(l): the arrival station for train l; (v) T_l: tonnage requirement of train l; (vi) β_l: horsepower per tonnage needed for train l; (vii) H_l: horsepower requirement of train l, which is defined as $H_l = \beta_l T_l$; and (viii) E_l: the penalty for using a single locomotive consist for train l.

Locomotive-Train Combinations. We need the following data for train-locomotive type combinations: (i) c_l^k: the cost incurred in assigning an active locomotive of type k to train l; (ii) d_l^k: the cost incurred in assigning a deadheaded locomotive of type k to train l; and (iii) t_l^k: the tonnage pulling capability provided by an active locomotive of type k to train l. Also specified for each train l are three disjoint sets of locomotive types: (i) *MostPreferred*[l], the preferred classes of locomotives; (ii) *LessPreferred*[l]: the acceptable (but not preferred) classes of locomotives; and (iii) *Prohibited*[l], the prohibited classes of locomotives. When assigning locomotives to a train, we can only assign locomotives from the classes listed as *MostPreferred*[l] and *LessPreferred*[l] (a penalty is associated for using *LessPreferred*[l]).

Constraints. We will now describe the constraints in the LSM. The constraints can be classified into two categories: *hard constraints* (which each locomotive assignment must satisfy) and *soft constraints* (which are highly desirable but not always required to be satisfied). We incorporate soft constraints by attaching a penalty for each violation of these constraints. The hard constraints for LSM are: (i) each train must be assigned locomotives with at least the required tonnage and horsepower; (ii) each train l is assigned locomotive types belonging to the set *MostPreferred*[l] and *LessPreferred*[l] only; (iii) each train must be assigned locomotives with at most 24 active axles; (iv) each train can be assigned at most 12 locomotives, including both the active and deadheaded locomotives; and (v) the number of assigned locomotives of each type is at most the number of available locomotives of that type. The soft constraints for LSM are: (i) consistency in the locomotive assignment (if a train runs five days a week, then it should be assigned the same consist each day it runs); (ii) consistency in train-to-train connections (if locomotives carrying a train to its destination station connect to another train originating at that station, then it should preferably make the same connection each day that both trains run); (iii) same-class connections (trains should connect to other trains in the same class); and (iv) avoiding consist busting as much as possible.

Objective Function. The objective function for the locomotive-scheduling model contains the following terms: (i) cost of ownership, maintenance, and fueling of locomotives; (ii) cost of active and deadheaded locomotives; (iii) cost of light-traveling locomotives; (iv) penalty for consist busting; (v) penalty for inconsistency in locomotive assignment and train-to-train connections; and (vi) penalty for using single-locomotive consists.

3. Space-Time Network

We will formulate the locomotive-scheduling problem as a multicommodity flow problem with side constraints on a network, which we call the *weekly space-time network*. Each locomotive type defines a commodity in the network. We denote the weekly space-time network as $G^7 = (N^7, A^7)$, where N^7 denotes the node set and A^7 denotes the arc set. Figure 1 displays a part of the weekly space-time network at one location. We construct the weekly space-time network as follows.

Nodes in the Weekly Space-Time Network. The network contains a *train arc* (i_l, j_l) for each train l. The tail node i_l of the arc denotes the event for the departure of train l at *dep-station*(l) and is called a *departure node*. The head node j_l denotes the arrival event of train l at *arr-station*(l) and is called an *arrival node*. For each arrival event, we create an *arrival-ground node*, and for each departure event, we create a *departure-ground* node. Each node is associated with two attributes: time and place. We use *time*(i) to denote the time attribute of node i in the weekly

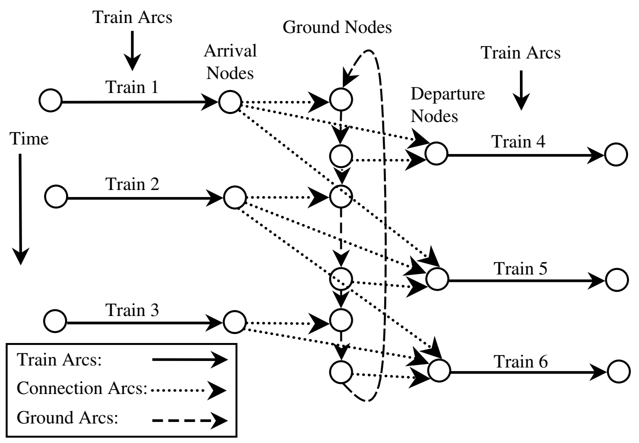

Figure 1 **A Part of the Weekly Space-Time Network**

space-time network. We denote the sets of departure, arrival, and ground nodes by the sets *DepNodes*, *ArrNodes*, and *GrNodes*, respectively. Let the set $AllNodes = DepNodes \cup ArrNodes \cup GrNodes$.

Arcs in the Weekly Space-Time Network. The network contains four types of arcs. The first is the set of train arcs, denoted by the set *TrArcs*, and contains an arc for every train. We connect each arrival node to the associated arrival-ground node by a directed arc called the *arrival-ground connection arc*. We connect each departure-ground node to the associated departure node through a directed arc called the *ground-departure connection arc*. We next sort all the 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*. (We assume without any loss of generality that ground nodes at each station have distinct time attributes.) The ground arcs allow inbound locomotives to stay in an inventory pool as they wait to be connected to the outbound trains. We also connect the last ground node in the week at a station to the first ground node of the week at that station through the ground arc; this ground arc models the ending inventory of locomotives for a week, which becomes the starting inventory for the following week. We also model the possibility of an inbound train sending its entire consist to an outbound train. We capture this possibility by creating *train-train connection arcs* from an arrival node to departure nodes whenever such a connection can be feasibly made. We also allow the possibility of light travel. We create a *light arc* in the weekly space-time network corresponding to each light travel possibility (to be described in detail in §6.4). Each light arc originates at a ground node (with a specific time and at a specific station) and also terminates at a ground node. Each light arc has a fixed charge that denotes the fixed cost of sending a single locomotive with crew from the origin of the light arc to its destination. We denote this fixed charge for a light-travel arc l by F_l. The light arc also has a variable cost that depends on the number of locomotives light traveling as a group. Therefore, the four kinds of arcs are: train arcs (*TrArcs*), connection arcs (*CoArcs*), ground arcs (*GrArcs*), and light-travel arcs (*LiArcs*). Let $AllArcs = TrArcs \cup CoArcs \cup GrArcs \cup LiArcs$.

We will formulate the locomotive-scheduling problem as a flow of different types of locomotives in the weekly space-time network. Locomotives flowing on train arcs are either active or deadheading; those flowing on light arcs are light traveling; and those flowing on connection and ground arcs are idling (that is, waiting between two consecutive assignments). We shall use the following additional notation for the weekly space-time networks in our MIP formulations: (i) $I[i]$: the set of incoming arcs into node $i \in AllNodes$; (ii) $O[i]$: the set of arcs emanating from node $i \in AllNodes$; (iii) F_l: the cost of an (active) light traveling locomotive with crew on a light arc $l \in LiArcs$; (iv) d_l^k: defined for every arc $l \in AllArcs$ (for a train arc l, d_l^k denotes the cost of deadheading of locomotive type k on train arc l and for every other arc it denotes the cost of traveling for a nonactive locomotive of locomotive type k on arc l); (v) *CB*: the set of all connection arcs from arrival nodes to ground nodes; alternatively, $CB = \{(i, j) \in AllArcs\colon i \in ArrNodes \text{ and } j \in GrNodes\}$; (vi) *CheckTime*: a time instant of the week when no event takes place (that is, no train arrives or departs at any station; we assume that *CheckTime* is Sunday midnight); and (vii) S: the set of arcs that cross the *CheckTime* (that is, $S = \{(i, j) \in AllArcs\colon time(i) < CheckTime < time(j)\}$).

4. The Mixed-Integer Programming Formulation

In this section, we present the mixed-integer programming (MIP) formulation of the locomotive-scheduling model. Our formulation has five sets of decision variables: (i) x_l^k: an integer variable representing the number of active locomotives of type $k \in K$ on the arc $l \in TrArcs$; (ii) y_l^k: an integer variable indicating the number of nonactive locomotives (deadheading, light-traveling, or idling) of type $k \in K$ on the arc $l \in AllArcs$; (iii) z_l: a binary variable that takes value 1 if at least one locomotive flows on the arc $l \in LiArcs \cup CoArcs$, and 0 otherwise; (iv) w_l: a binary variable that takes value 1 if there is a flow of a single locomotive on arc $l \in TrArcs$, and 0 otherwise; and (v) s^k: an integer variable indicating the number of unused locomotives of type $k \in K$. It terms of these variables, we next provide the MIP formulation of the LSM.

$$\min z = \sum_{l\in TrArcs}\sum_{k\in K} c_l^k x_l^k + \sum_{l\in AllArcs}\sum_{k\in K} d_l^k y_l^k + \sum_{l\in LiArcs} F_l z_l + \sum_{l\in CB} V z_l + \sum_{l\in TrArcs} E_l w_l - \sum_{k\in K} G^k s^k \tag{1a}$$

subject to:

$$\sum_{k\in K} t_l^k x_l^k \geq T_l, \quad \text{for all } l\in TrArcs, \tag{1b}$$

$$\sum_{k\in K} h^k x_l^k \geq \beta_l T_l, \quad \text{for all } l\in TrArcs, \tag{1c}$$

$$\sum_{k\in K} \lambda^k x_l^k \leq 24, \quad \text{for all } l\in TrArcs, \tag{1d}$$

$$\sum_{k\in K} (x_l^k + y_l^k) \leq 12, \quad \text{for all } l\in TrArcs \cup LiArcs, \tag{1e}$$

$$\sum_{l\in I[i]} (x_l^k + y_l^k) = \sum_{l\in O[i]} (x_l^k + y_l^k), \quad \text{for all } i\in AllNodes, \text{for all } k\in K, \tag{1f}$$

$$\sum_{k\in K} y_l^k \leq 12 z_l, \quad \text{for all } l\in CoArcs\cup LiArcs, \tag{1g}$$

$$\sum_{l\in O[i]} z_l = 1, \quad \text{for all } i\in ArrNodes, \tag{1h}$$

$$\sum_{l\in I[i]} z_l = 1, \quad \text{for all } i\in DepNodes, \tag{1i}$$

$$\sum_{k\in K} (x_l^k + y_l^k) + w_l \geq 2, \quad \text{for all } l\in TrArcs, \tag{1j}$$

$$d\sum_{l\in S} (x_l^k + y_l^k) + s^k = B^k, \quad \text{for all } k\in K, \tag{1k}$$

$$x_l^k, y_l^k \geq 0 \text{ and integer}, \quad \text{for all } l\in TrArcs, \text{for all } k\in K, \tag{1l}$$

$$z_l \in \{0,1\}, \quad \text{for all } l\in CoArcs\cup LiArcs, \tag{1m}$$

$$w_l \in \{0,1\}, \quad \text{for all } l\in TrArcs. \tag{1n}$$

We will now explain the above formulation to confirm that it correctly represents the LSM. We will first discuss the constraints. Constraint (1b) ensures that the locomotives assigned to a train provide the required tonnage, and constraint (1c) ensures that the locomotives assigned provide the required horsepower. Constraint (1d) models the constraint that the number of active axles assigned to a train does not exceed 24; the constraint (1e) models the constraint that every train arc and light arc are assigned at most 12 locomotives. The flow balance constraint (1f) ensures that the number of incoming locomotives equals the number of outgoing locomotives at every node of the weekly space-time network. Constraint (1g) makes the fixed-charge variable z_l equal to 1 whenever a positive flow takes place on a connection arc or a light arc; this constraint also ensures that no more than 12 locomotives flow on any light arc. Constraint (1h) states that, for each inbound train, all the inbound locomotives use only one connection arc; either all the locomotives go to the associated ground node (in which case consist busting takes place), or all the locomotives go to another outbound train (in which case consist busting does not take place and there is a train-to-train connection). Constraint (1i) states that for each outbound train all the outbound locomotives either come from a ground node or all the locomotives come from an incoming train. Constraint (1j) makes the variable w_l equal to 1 whenever a single-locomotive consist is assigned to train l. Finally, constraint (1k) counts the total number of locomotives used in the week; this is the sum of the flow of locomotives on all the arcs crossing the *CheckTime*. The difference between the number of locomotives available and the number of locomotives used equals the number of unused locomotives (s^k).

We will now discuss the objective function (1a), which contains six terms. The first term denotes the cost of actively pulling locomotives on train arcs. The second term captures the cost of deadheading locomotives on train arcs and light-travel arcs, as well as the cost of idling locomotives. We also include the variable cost of consist busting in the definition of the term d_l^k for each arc $l \in CB$. The third term, $\sum_{l\in LiArcs} F_l z_l$, denotes the fixed cost of light-traveling locomotives. The fourth term, $\sum_{l\in CB} V z_l$, denotes the fixed cost of consist busting. The fifth term, $\sum_{l\in TrArcs} E_l w_l$, denotes the penalty associated with the single-locomotive consists; and the sixth term, $\sum_{k\in K} G^k s^k$, represents the savings accrued from not using all the locomotives. Observe that we can obtain different levels of consist busting by selecting different values of the consist busting cost V. The greater the value of V, the less the amount of consist busting in an optimal solution.

Observe that formulation (1) assumes that any locomotive type can flow on any train arc. However, recall from our discussion in §2 that each train arc l has *MostPreferred*$[l]$, *LessPreferred*$[l]$, and *Prohibited*$[l]$ sets of locomotives. We handle these constraints in the following manner. To formulation (1) we add the constraints $x_l^k = 0$ for each $k \in Prohibited[l]$ and each $l \in TrArcs$. These constraints ensure that prohibited locomotives are never used on train arcs. To discourage the flow of locomotive types belonging to the *LessPreferred* sets, we multiply c_l^k by a suitable parameter larger than 1 (for example, 1.2) for each $k \in LessPreferred[l]$ and each $l \in TrArcs$.

Our formulation (1) incorporates all the hard constraints described in §2, but not all the soft constraints, such as the consistency constraints. Including those constraints would make the formulation unmanageably large.

The locomotive-scheduling problem (1) can be shown to be NP-complete. We prove this result by reducing a well-known NP-complete problem, 3-partition, into a locomotive-scheduling problem (Garey and Johnson 1979). We will prove this result next.

3-Partition Problem. Given $3n$ integers $a_1, a_2, \ldots, a_{3n}$, and $b = \sum_{i=1}^{3n} a_i/n$, where $a_i > b/4$ for all $i = 1, 2, \ldots, 3n$, does there exist a partition of $\{1, \ldots, 3n\}$ into n subsets $S_1, S_2, \ldots, S_n$ such that $\sum_{i\in S_j}^{3n} a_i = b$ for all $j = 1$ to n?

Suppose that there are exactly n trains, each with a horsepower requirement of b units, and they all travel from the same origin city to the same destination city. Suppose further that there are $3n$ locomotives, where locomotive i provides horsepower a_i. Assume that $b = \sum_{i=1}^{3n} a_i/n$, where $a_i > b/4$ for all $i = 1, 2, \ldots, 3n$. Then the locomotive-scheduling problem has a feasible solution if and only if the 3-partition problem has a feasible solution. Thus, the 3-partition can be solved by solving a locomotive-scheduling problem.

In the data provided to us by CSX, there were 538 trains, each of which operated several days in a week, and there were five locomotive types. The weekly space-time network consisted of 8,798 nodes and 30,134 arcs. The MIP formulation (1) consisted of 197,424 variables and 67,414 constraints. This formulation could not be solved to optimality or near optimality using the commercial software CPLEX 7.0. As a matter of fact, we were unable to solve even the linear programming (LP) relaxation of the problem. Additionally, this formulation does not capture the consistency constraints incorporating which would even further expand the size of the problem. We therefore developed an alternative approach that simultaneously helped enforce consistency while dramatically reducing the problem size. We will describe our approach in the next section.

5. Simplifying the Model

We analyzed the number of CSX trains running with different frequencies in one week. The table shown in Figure 2 provides these numbers. The first column in the table gives the weekly train frequency, or how often the trains run in a week. The second column in the table lists the number of trains in the train schedule that run with the frequency given in the first column. The third column is a product of the first and second columns; the fourth column provides a cumulative sum of the third column. The fifth column expresses the values in the fourth column in percentage notation. Observe that the third column lists the number of train arcs in the weekly space-time network for trains with different frequencies. The table reveals that about 94% of the train arcs in the space-time network correspond to the trains that run five, six, or seven days.

We now make an approximation that reduces the size of the MIP substantially and also helps us satisfy the consistency constraints. We create a daily locomotive-scheduling model that is a simplification of the weekly locomotive-scheduling model in the following manner. We assume that: (i) all trains that run p days or more per week run *every* day of the week; and (ii) all trains that run fewer than p days do not run at all. This assumption results in an approximation in the sense that we provide locomotives to the trains that do not exist and do not provide locomotives to trains that do exist. For the train data given in Table 1, Table 2 gives the number of such trains for different values of p, as follows.

To transform the solution of the daily locomotive-scheduling solution into a feasible solution for the weekly scheduling problem, we take locomotives from the trains that exist in the daily problem but do not exist in the weekly problem (type 1 trains) and assign them to the trains that do not exist in the daily problem, but exist in the weekly problem (type 2 trains). Also, we possibly use additional locomotives to meet the constraints. Clearly, we can convert the solution of the daily problem into a weekly problem more effectively if the number of type 1 trains are less than the number of type 2 trains, but still as close as possible. From this perspective, we find that $p = 5$ is the best choice for p, which we also determined in our computational investigations. We therefore solve the locomotive-scheduling problem in two stages. The first stage solves the daily locomotive-scheduling problem, and the second stage modifies the daily locomotive schedule to obtain the weekly locomotive schedule.

Table 1 Analysis of Trains and Their Frequencies

Train frequency (A)	Number of trains (B)	$A*B$	Cumulative sum of $A*B$	Cumulative percentage of $A*B$ (%)
7	372	2,604	2,604	78.3
6	62	372	2,976	89.5
5	29	145	3,121	93.9
4	24	96	3,217	96.8
3	20	60	3,277	98.6
2	16	32	3,309	99.5
1	15	15	3,324	100

Table 2 Choosing the Right Value of p

p	1	2	3	4	5	6	7
Number of nonexisting trains that are assigned locomotives (type 1 trains)	442	352	272	192	120	62	0
Number of existing trains that are not assigned locomotives (type 2 trains)	0	15	47	107	203	348	720

6. Solving the Daily Locomotive Scheduling Problem

In this section, we will describe how to solve the daily scheduling problem, including: (i) how to construct the daily space-time network and formulate the MIP for the daily scheduling problem; and (ii) how to solve the daily locomotive-scheduling problem using a decomposition-based approach.

6.1. Constructing the Daily Space-Time Network and Formulating the MIP

The daily space-time network is constructed for the daily locomotive-scheduling problem for which each train is assumed to run each day of the week. It is constructed similarly to how we construct the weekly space-time network. We represent the daily space-time network as $G^1 = (N^1, A^1)$. The daily space-time network is about seven times smaller than the weekly space-time network.

In the daily model, the departure time is the number of minutes past midnight that the train departs, and the arrival time is the number of minutes past midnight that the train arrives. The day of the week does not play a role in this daily model. For a train l, let $day(l)$ denote the number of times the train crosses the midnight time line as it goes from its origin to its destination. For example, consider a train that leaves station A at 7 AM on one day and arrives at 4 PM two days later at station B. For example, the train starts on Monday at 7 AM and arrives at 4 PM on Wednesday. In this case, $day(l) = 2$. If this train were to repeat each day in our weekly model, then two copies of the train would cross the time line at midnight on Sunday: the train that starts at 7 AM on Saturday and ends at 4 PM on Monday, and the one that starts at 7 AM on Sunday and ends at 4 PM on Tuesday. In general, if $day(l) = m$ in our daily model, then there are m copies of the train l that cross the midnight demarcation at Sunday time line in our weekly model. Therefore, assigning a locomotive in our daily model to a train l with $day(l) = m$ corresponds to assigning m locomotives of the same type in the weekly model.

To account for the impact of $day(l)$, we modify the fleet size constraint of the locomotive-scheduling problem as follows:

$$\sum_{l \in S} (x_l^k + y_l^k) day(l) + s^k = B^k, \quad \text{for all } k \in K.$$

The rest of the formulation of the locomotive-scheduling problem (1) on the daily space-time network is identical to the one provided earlier.

6.2. Solving the MIP for the Daily Scheduling Problem

Though the space-time network for the daily scheduling problem was substantially smaller than the space-time network for the weekly scheduling problem, we found it to be too large to be solved to optimality or near optimality. The daily locomotive-scheduling problem consisted of 463 train arcs, and the daily space-time network contained 1,323 nodes and 30,034 arcs. The MIP formulation consisted of 22,314 variables and 9,974 constraints. The LP relaxation took a few seconds to solve, but the MIP did not provide any integer feasible solution in 72 hours of running time. We conjecture that the biggest source of difficulty was the presence of fixed-charge variables z_l (for connection and light arcs), which takes value 1 whenever there is a positive flow on arc l. It is well known that MIPs with many fixed-charge variables often produce weak lower bounds.

To obtain high-quality feasible solutions and to maintain a relatively small total running time of the algorithm, we decided to eliminate the fixed-charge variables from the MIP formulation using heuristics. The heuristics also allowed us greater flexibility in determining what kind of solution we want. In our formulation, we have two kinds of fixed-charge variables: one corresponding to connection arcs and the other corresponding to light arcs. We will first consider the fixed-charge variables corresponding to the train-train connection arcs, followed by the fixed-charge variables corresponding to light arcs. We will also consider some cases in which fixed-charge variables can be eliminated without any loss of generality.

6.3. Determining Train-Train Connections

Train companies often specify some "hardwired" train-train connections, that is, they specify some inbound trains whose consist must go to the specified outbound trains. These hardwired train-train connections are easy to enforce in the space-time network. Sometimes, a station has a unique inbound train and a unique outbound train. If this station has no light-traveling locomotives coming into it, then all the locomotives brought in by the inbound train will be automatically assigned to the outbound train. Hence, there will be no consist busting and there will be no need for fixed-charge variables. Sometimes, train companies also have rules that control which train-train connections are permissible and/or desirable. We can eliminate such inadmissible train-train connections easily. Among the admissible train-train connections, we fix several of them using an iterative process, which we describe next.

We begin with a daily space-time network that contains all the candidate train-train connection arcs and candidate light arcs. We next solve the linear programming relaxation of the locomotive-scheduling model for which there are no fixed-charge variables and constraints (that is, constraints (1g) through (1i)), and all the train-to-ground and ground-to-train connections have a large cost to discourage the flow on such arcs (or, alternatively, to discourage consist busting). Let $\varphi(l)$ denote the total flow of locomotives (of all types) on any arc l in the daily space-time network. We next select a candidate train-train connection arc h with the largest value of $\varphi(h)$. This arc indicates a successful potential train-train connection. We make this connection arc the unique connection arc for the two corresponding trains and resolve the linear programming relaxation. If this linear programming relaxation is infeasible or if it increases the cost of the new solution by an amount greater than θ, we do not make this train-train connection; otherwise, we keep this connection. In any case, we remove arc h from the list of candidate train-train connections. We next select another candidate train-train connection arc h' with the largest value of $\varphi(h')$ in the current flow solution and repeat this process until either we have reached the desired number of train-train connections (as specified by some parameter γ), or until the set of candidate train-train connections becomes empty.

Our method is parameter driven. By suitably identifying the values of the two parameters θ and γ, we can govern the behavior of the algorithm. By choosing the higher value of θ, we can increase the number of train-train connections. We can also increase the number of train-train connections by increasing the value of the parameter γ. We illustrate in §8 that by using this heuristic technique, we were able to increase the number of train-train connections to 72%, which was substantially higher than the goals originally set by our industry sponsor. In fact, 72% was originally not considered to be an achievable goal. The parameters θ and γ need to be assigned correct values so we can determine the desired consist busting. We used $\theta = \$1{,}000$ and varied γ to get different levels of consist bustings.

6.4. Determining Light-Travel Arcs

Light travel of locomotives plays an important role in locomotive scheduling and can substantially impact the quality of the solution. Incorporating light travel in the locomotive-scheduling model requires two decisions: (i) *candidate light arcs* (what the light travel arcs should be in the space-time network); and (ii) *flow on light-travel arcs* (which locomotives should flow on the candidate light arcs). In principle, we could allow light travel of locomotives from any station to any station at any time of the day. To capture all these possibilities of light travel in our network, we need to add a large number of arcs in the daily space-time network. This, however, would increase the size of the network substantially as well as the number of flow variables. Instead of introducing a large number of light-travel arcs, we developed a heuristic procedure to create a small but potentially useful collection of light-travel arcs; we developed another heuristic procedure for selecting a subset of these arcs for light travel. We will now describe these procedures in greater detail.

Our first procedure determines the candidate light-travel possibilities. For each such arc, we need to decide its origin station, its start time, its destination station, and its end time. The end time can be computed from the origin station, its start time, and its destination station if we know the average speed of the light-traveling locomotives. In the railroad network, light-traveling locomotives typically travel from "power sources" (stations where inbound trains require substantially more tonnage/horsepower than the outbound trains) to "power sinks" (stations where outbound trains require substantially more tonnage/horsepower than the inbound trains). First, we construct the *space network*, which is the same as the daily space-time network described in §6.1 except that we ignore the time element. We next determine the additional availability of pulling power at power sources and the additional demand for pulling power at power sinks in the space network of our weekly model. To incorporate both horsepower and tonnage constraints simultaneously, we translate this information into the number of locomotives of a standard type. This gives us node imbalances of locomotives in the space network. We then solve a minimum-cost flow problem (see, for example, Ahuja, Magnanti, and Orlin 1993) in the space network to determine the optimal locomotive flow in the network. If there is a positive flow of locomotives on arc (i, j) in the space network greater than some threshold value (say, two locomotives), then we create candidate light-travel arcs from station i to station j in the space-time network departing from city i every eight hours. For the data provided by CSX, we found 58 arcs in the space network with a flow of greater than two locomotives and we created 174 candidate light arcs in the space-time network.

In our formulation described before, we need to create a fixed-charge variable for each candidate light arc, and we cannot solve the formulation for this many fixed-charge variables in the required time. Our second procedure eliminates the fixed-charge variables. The procedure works by first solving a linear programming formulation of the locomotive-scheduling problem in which all the candidate light arcs are added. It then removes all those light arcs that have zero flow. Let $\varphi(l)$ denote the total number of locomotives flowing on a candidate light arc l. We then perform the following iterative step. We select the candidate light arc l with the smallest value of flow $\varphi(l)$. Suppose this is arc r. We delete arc r from the network and solve the LP relaxation of the locomotive-scheduling problem. If the deletion of this candidate light arc causes the total cost of flow (of the LP relaxation) to increase by at least η units, we keep this arc and remove it from the candidate light arcs list; otherwise, we permanently delete this arc. We repeat this process until there are no unexamined candidate light arcs. The number of light arcs generated by this module crucially depends upon the value of the parameter η. If we increase the value of η, then we increase the number of arcs deleted from the network, which will result in fewer light arcs being generated. Hence, by assigning suitable values to this parameter, one can generate an appropriate number of light arcs. We used $\eta = \$1{,}000$ in our implementation.

6.5. Determining Active and Deadheaded Locomotive Flow Variables

As described before, the use of heuristics to determine train-train connections and light-travel arcs eliminates the fixed-charge variables. We next determine the remaining variables that are the active and deadheaded locomotives on arcs in the network. To determine these variables, we solve the following integer programming problem:

$$\min\ z = \sum_{l\in TrArcs}\sum_{k\in K} c_l^k x_l^k + \sum_{l\in AllArcs}\sum_{k\in K} d_l^k y_l^k + \sum_{l\in TrArcs} E_l w_l - \sum_{k\in K} G^k s^k \tag{2a}$$

subject to:

$$\sum_{k\in K} t_l^k x_l^k \ge T_l, \quad \text{for all } l \in TrArcs, \tag{2b}$$

$$\sum_{k\in K} h^k x_l^k \ge \beta_l T_l, \quad \text{for all } l \in TrArcs, \tag{2c}$$

$$\sum_{k\in K} \lambda^k x_l^k \le 24, \quad \text{for all } l \in TrArcs, \tag{2d}$$

$$\sum_{k\in K} (x_l^k + y_l^k) \le 12, \quad \text{for all } l \in TrArcs \cup LiArcs, \tag{2e}$$

$$\sum_{l\in I[i]} (x_l^k + y_l^k) = \sum_{l\in O[i]} (x_l^k + y_l^k), \quad \text{for all } i \in AllNodes, \text{ for all } k \in K, \tag{2f}$$

$$\sum_{k\in K} (x_l^k + y_l^k) + w_l \ge 2, \quad \text{for all } l \in TrArcs, \tag{2g}$$

$$\sum_{l\in S} (x_l^k + y_l^k)\, day(l) + s^k = B^k, \quad \text{for all } k \in K, \tag{2h}$$

$$x_l^k, y_l^k \ge 0 \text{ and integer}, \quad \text{for all } l \in TrArcs, \text{ for all } k \in K, \tag{2i}$$

$$w_l \in \{0, 1\}, \quad \text{for all } l \in TrArcs. \tag{2j}$$

We refer to a solution of (2) by (x, y), because w can be deduced using the solution (x, y). This integer programming model is much easier to solve than the model described in §6.1. It does not contain fixed-charge variables and has fewer locomotive flow variables. The locomotive-scheduling problem we solved had 2,315 x_l^k variables; 6,120 y_l^k variables; 463 w_l variables; and 7,782 constraints. We solved (2) using CPLEX 7.0 and set the CPLEX parameters so that a very good integer solution is obtained in the early part of the branch-and-bound algorithm. We found that CPLEX 7.0 provided a high-quality solution within 15 minutes of execution time, but it did not terminate even when it was allowed to run for over 48 hours. We also found that the algorithm, when run for 24 hours, was not much better than the best integer solution obtained within 15 minutes; therefore, prematurely terminating the algorithm after 15 minutes did not much affect the quality of the solution obtained.

6.6. Neighborhood Search Algorithm

The integer solution obtained by the integer programming software CPLEX 7.0 is not, in general, an optimal solution for the locomotive-scheduling problem. We found that this solution can be easily improved by a modest modification of the solution. We used a neighborhood search algorithm to look for possible improvements. A neighborhood search algorithm typically starts with an initial feasible solution and repeatedly replaces it by an improved neighbor until we obtain a solution that is at least as good as its neighbors. At this point, the current solution is called a local optimal solution (Aarts and Lenstra 1997). We call a neighborhood search algorithm a *very large-scale neighborhood* (*VLSN*) *search algorithm* if the size of the neighborhood is very large, possibly exponential in terms of the input size parameters, and we use implicit enumeration algorithms to identify an improved neighbor. We refer the reader to the papers by Ahuja et al. (2001a, b; 2002) for additional details on VLSN search algorithms. This neighborhood can also be considered a specific implementation of the concept of *referent domain optimization*, as it is briefly described in Glover and Laguna (1997). We define a neighborhood using an integer programming formulation and identify an improved neighbor with the software CPLEX.

Let $(\bar{x}, \bar{y})$ be a feasible solution of (2). We call a solution (x, y) a *neighbor* of $(\bar{x}, \bar{y})$ if (x, y) is feasible for (2b)–(2j) and it differs from $(\bar{x}, \bar{y})$ for one locomotive type only; that is, $\bar{x}^q = x^q$ and $\bar{y}^q = y^q$ for all $q \in K\backslash\{k\}$ for some locomotive type k. In other words, all the feasible locomotive flows that can be obtained by changing the locomotive flow for one locomotive type only define the neighborhood of the solution (x, y). Mathematically, all the solutions of the following integer program define the neighborhood of the solution $(\bar{x}, \bar{y})$.

$$\text{minimize}\, z^k = \sum_{l\in TrArcs} c_l^k x_l^k + \sum_{l\in AllArcs} d_l^k y_l^k + \sum_{l\in TrArcs} E_l w_l - G^k s^k \tag{3a}$$

subject to:

$$t_l^k x_l^k \geq T_l - \sum_{q\in K\setminus\{k\}} \bar{x}_l^q, \quad \text{for all } l \in TrArcs, \tag{3b}$$

$$h^k x_l^k \geq \beta_l T_l - \sum_{q\in K\setminus\{k\}} h^q \bar{x}_l^q, \quad \text{for all } l \in TrArcs, \tag{3c}$$

$$\lambda^k x_l^k \leq 24 - \sum_{q\in K\setminus\{k\}} \lambda^q \bar{x}_l^q, \quad \text{for all } l \in TrArcs, \tag{3d}$$

$$x_l^k + y_l^k \leq 12 - \sum_{q\in K\setminus\{k\}} (\bar{x}_l^k + \bar{y}_l^k), \quad \text{for all } l \in TrArcs \cup LiArcs, \tag{3e}$$

$$\sum_{l\in I(i)} (x_l^k + y_l^k) = \sum_{l\in O(i)} (x_l^k + y_l^k), \quad \text{for all } i \in AllNodes, \tag{3f}$$

$$x_l^k + y_l^k + w_l \geq 2 - \sum_{q\in K\setminus\{k\}} (\bar{x}_l^q + \bar{y}_l^q), \quad \text{for all } l \in TrArcs, \tag{3g}$$

$$\sum_{l\in AllArcs} (x_l^k + y_l^k) day(l) + s^k = B^k, \tag{3h}$$

$$x_l^k, y_l^k \geq 0 \text{ and integer}, \quad \text{for all } l \in TrArcs, \tag{3i}$$

$$w_l \in \{0, 1\}, \quad \text{for all } l \in TrArcs. \tag{3j}$$

We call the solution $(\bar{x}, \bar{y})$ a local optimal solution if $(\bar{x}, \bar{y})$ is an optimal solution of (3) for each $k \in K$. Thus, a solution $(\bar{x}, \bar{y})$ is a local optimal solution of the locomotive-scheduling problem if each single locomotive type is optimally scheduled when the schedule of other locomotive types is not allowed to change. Although (3) is an integer programming problem with 2,168 variables and 3,437 constraints, CPLEX was able to solve it to optimality within a few seconds.

To summarize, we take the solution provided by the integer programming software for the daily scheduling problem (3) as the starting solution of our neighborhood search algorithm and solve a sequence of problems (3) for all locomotives of type $q \in K$. We stop when the solution cannot be improved for any locomotive type. The solution at this stage is a local optimal solution.

7. Solving the Weekly Locomotive-Scheduling Problem

We will now describe how we solve the weekly locomotive-scheduling problem. Recall from our discussion in §5 that to handle the consistency constraints and to keep the problem size manageable, we solve the problem in two stages. The first stage assumes that all trains run every day of the week. To satisfy this assumption, we eliminate trains that run fewer than p days (for example, $p = 5$), and all other trains are assumed to run every day of the week. When we apply the solution of the daily scheduling to the weekly scheduling problem by repeating it every day, then we provide locomotives to some trains that do not exist and do not provide locomotives to those trains that do exist. To transform this solution into a feasible and effective solution for the weekly scheduling problem, we take locomotives from the trains that exist in the daily problem but do not exist in the weekly problem and assign them to the trains that do not exist in the daily problem but exist in the weekly problem; we possibly use additional locomotives to meet the constraints. In this section, we will describe this method in greater detail.

First, we will define some notation in order to simplify the presentation. There are two sets of trains in the locomotive-scheduling problem: (i) Φ, the set of trains that operate p or more days a week; and Ψ, the set of trains that operate fewer than p days a week. Let Φ^1 denote the set of train arcs corresponding to the trains in Φ in the daily space-time network G^1, and let Φ^7 denote the set of train arcs corresponding to the trains in Φ in the weekly space-time network G^7. Observe that each arc in Φ^1 induces p or more corresponding arcs in Φ^7. Let Ψ^7 denote the set of train arcs corresponding to the trains in Ψ in the weekly space-time network G^7. Observe that trains in Ψ do not have any corresponding train arc in G^1. Also observe that in the weekly space-time network, $TrArcs = \Phi^7 \cup \Psi^7$.

Suppose that the optimal solution for the daily locomotive-scheduling problem produces the following solution:

$\bar{x}_l^k$: The number of active locomotives of type k assigned to train arc $l \in \Phi^1$.

$\bar{y}_l^k$: The number of deadheaded locomotives of type k assigned to train arc $l \in \Phi^1$.

We have the following objectives for the weekly locomotive-scheduling problem:

(i) Each arc $l \in \Phi^7$ has a corresponding arc in Φ^1, say, $\bar{l}$. The locomotive assignment of arc $\bar{l} \in \Phi^1$ in the daily locomotive-scheduling problem becomes the "target flow" for the corresponding arc $l \in \Phi^7$ in the weekly locomotive-scheduling problem. We would like the locomotive flow on each arc in Φ^7 to be as close to its target flow as possible, as this ensures that the weekly locomotive schedule is consistent. Changes to the target flow can be made, but they are penalized, as changes may affect the consistency of the solution. Henceforth, we assume that $\bar{x}_l^k$ and $\bar{y}_l^k$ denote the target flow for each arc $l \in \Phi^7$.

(ii) Each arc $l \in \Psi^7$ should be assigned a sufficient number of locomotives so that its horsepower and tonnage requirements are satisfied. Observe that we do not consider consistency for train arcs in Ψ^7. Only 6% of the train arcs are in Ψ^7, and ignoring consistency of these arcs does not have a major impact on the overall consistency of the solution.

(iii) If there is a train-train connection from train A to train B in the daily space-time network, then the same train-train connection should be carried over to the weekly space-time network on all the days when both trains A and B run.

The weekly space-time network is 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 train-train connection arcs. For a train-train connection arc in the daily space-time network, we have a corresponding train-train connection arc only on those days when both the associated trains run, and not otherwise.

We will now discuss how to determine a locomotive schedule in the weekly space-time network. We first formulated this problem as an integer programming problem, which is similar to the IP formulation (1) presented in §4. However, we were unable to solve it using CPLEX 7.0. The integer programming problem for the weekly space-time network is about seven times larger than the corresponding problem for the daily space-time network, and CPLEX ran for 72 hours without obtaining any integer feasible solution. Because our goal was to solve the locomotive-scheduling problem within 30 minutes of computational time, we needed to follow another approach.

We next attempted to determine the locomotive schedule one locomotive type at a time. This converted a multicommodity flow problem (with each locomotive type defining a commodity) with side constraints into a sequence of single-commodity flow problems with side constraints, one for each locomotive type. We found that CPLEX 7.0 efficiently solved these single-commodity flow problems with side constraints. We will now describe this approach in greater detail.

We first arranged different locomotive types in order and then, following this order, considered them one by one.

Suppose we are considering locomotive type k. Our goal is to determine the schedule of locomotive type k so as to:

$$\text{minimize} \sum_{l\in\Psi}(c_l^k - M)x_l^k + \sum_{l\in\Phi} c_l^k x_l^k + \sum_{l\in AllArcs} d_l^k x_l^k + \sum_{l\in TrArcs} E_l w_l - G^k s^k + P\sum_{l\in\Phi}(\alpha_l^{+k} + \alpha_l^{-k}). \quad (4a)$$

Subject it to (3b)–(3g), (3i)–(3j), and the following constraints:

$$x_l^k \le U_l^k \quad \text{for all } l \in \Psi, \quad (4b)$$

$$\alpha_l^{+k} - \alpha_l^{-k} = (x_l^k + y_l^k) - (\bar{x}_l^k + \bar{y}_l^k) \quad \text{for all } l \in \Phi, \quad (4c)$$

$$\alpha_l^{+k}, \alpha_l^{-k} \ge 0 \text{ and integer}, \quad \text{for all } l \in TrArcs, \quad (4d)$$

$$\sum_{l\in S}(x_l^k + y_l^k) + s^k = B^k, \quad \text{for all } k \in K, \quad (4e)$$

where M is a large positive number, and where

$$U_l^k = \max\left\{\frac{T_l - \sum_{q\in K\setminus\{k\}} t_l^q \bar{x}_l^q}{t_l^k}, \frac{\beta_l T_l - \sum_{q\in K\setminus\{k\}} h_l^q \bar{x}_l^q}{h^k}\right\}$$

denotes the number of active locomotives of type k needed to meet the tonnage and horsepower requirements of the train arc l.

This formulation is similar to formulation (3), which finds the optimal flow of locomotives of type k while keeping the flow of other locomotive types intact. In addition, the formulation attempts to send the sufficient number of locomotives on arcs in Ψ; this is achieved by subtracting a large positive number M from the active locomotive costs of arcs in Ψ in the objective function (4a). The formulation also attempts to find a flow which is "close" to the target flow on the arcs in Φ; this is accomplished by measuring the positive or negative deviation (α_l^{+k} or α_l^{-k}) from the target flow through constraints (4c) and (4d) and penalizing it in the objective function (4a). By assigning a suitable value to the coefficient P, we make the locomotive flow (x^k, y^k) sufficiently close to the target flow on the arcs in the set Φ. In our computational investigations, we used $P = \$1{,}000$.

We solved formulation (4) multiple times and with different objective functions to obtain improved solutions. For example, our primary objective is to provide a sufficient number of locomotives to the trains in Ψ. We accomplish our goal by choosing a sufficiently large value of M (say, 10^6), and solving (4) for each locomotive type one at a time. The order in which we consider different locomotive types is important. We consider the locomotive types in the decreasing order of their availability; that is, the locomotive type that has the largest number of locomotives available is considered first, followed by the locomotive type that has the second-largest number of locomotives available, and so on. The intuition behind this rule is that if some trains need locomotives, we will try to power them by the locomotive type that has a greater availability; this rule tends to generate feasible locomotive assignments. Once all trains have a sufficient number of active locomotives, we set $M = 0$ and solve it again for all locomotive types. Depending on the value of P, the optimal solution balances the total flow cost and the penalty for deviation from the target flow. We may need to experiment with different values of P before we obtain the right balance between the two terms.

For our data, formulation (4) is defined over a network with 8,798 nodes and 9,139 arcs. The formulation (4) consists of 15,266 variables and 13,142 constraints. CPLEX 7.0 was able to solve it for one locomotive type in a few seconds. We took a total of two to three minutes to obtain a satisfactory solution in the weekly space-time network. Whereas the integer programming formulation of the weekly space-time network was insolvable, solving it heuristically using the constrained network flow algorithm was quite efficient.

After we obtained a feasible solution of the weekly locomotive-scheduling problem, we applied a neighborhood search algorithm to it. We considered each locomotive type one by one and tried to determine the optimal flow of the selected locomotive type while keeping the flow of other locomotive types intact. This subproblem is a (single-commodity) constrained minimum cost flow problem and can be very efficiently solved. We terminate when the solution value does not improve for any locomotive type. Observe that this algorithm can also be regarded as a very large-scale neighborhood search algorithm and is a modification of the neighborhood search algorithm described in §6.6 for the daily space-time network. Finally, after we have obtained the solution of the weekly locomotive-scheduling problem, we can create additional train-train connections by matching the locomotive assignments of inbound and outbound trains at each station if they have the same consist. For the data provided by CSX, we were able to create about 10%–15% additional train-train connections with the use of this algorithm.

8. Computational Results

In this section, we will present our computational results. CSX Transportation supplied us with data files for several scenarios. Here we present the results for three scenarios corresponding to three months. These scenarios contained 3,324 trains originating from and terminating at 119 stations; however, the tonnage and horsepower requirements of these trains were different in different scenarios. 3,316 locomotives belonged to five locomotive types that we could use for scheduling. All of our algorithms were implemented in C++, and we made extensive use of callable libraries in CPLEX 7.0. The algorithms were run and tested on a Pentium III PC with a speed of 750 MHz. We call our software the *Advanced Locomotive Scheduling* (*ALS*) system. Our algorithms have been very consistent in terms of their running-time requirements and the reduction in locomotives.

Table 3a Comparison of ALS and CSX Solutions: The Number of Locomotives of Different Types Used by the CSX and ALS Software

Type of locomotives	Scenario 1		Scenario 2		Scenario 3	
	CSX	ALS	CSX	ALS	CSX	ALS
SD40	498	249	519	283	550	323
SD50	171	160	162	138	174	161
C40	621	487	619	466	620	432
AC44	164	154	155	154	155	155
AC60	160	160	160	160	160	160
Total	1,614	1,210	1,615	1,201	1,659	1,231

Table 3(a) compares the solutions obtained by ALS with those of CSX's software. In each of these cases, the ALS solution is substantially superior to the CSX solution; ours reduces the total cost substantially and dramatically decreases the number of locomotives used. The ALS solution used 350–400 fewer locomotives than the CSX solution with medium consist busting (50%) and medium light travel (0.8%). Recall from §1 that CSX handles consist assignment and locomotive scheduling separately and that their locomotive-scheduling phase considers each locomotive type individually. We achieved the dramatic decrease in the number of locomotives mainly by combining consist assignment and locomotive scheduling and by considering all locomotive types together instead of individually.

Integrating consist assignment and locomotive scheduling also significantly reduced the fraction of locomotives that deadhead. Table 3(b) presents the statistics on the percentage of the times when locomotives are actively pulling trains, deadheading on trains, light traveling, or idling at stations (for maintenanceor just waiting to be assigned to out-

Table 3b Comparison of ALS and CSX Solutions: Comparison of Other Statistics

Performance measure	Scenario 1		Scenario 2		Scenario 3		Average	
	CSX (%)	ALS (%)	CSX (%)	ALS (%)	CSX (%)	ALS (%)	CSX (%)	ALS (%)
Active time	31.30	44.40	31.30	44.95	29.20	47.03	30.60	45.46
Deadheading time	19.60	8.10	19.60	9.31	20.18	8.31	19.79	8.57
Idling time	49.10	46.70	49.10	45.00	50.62	43.78	49.61	45.16
Light-traveling time	0	0.80	0.00	0.74	0.00	0.88	0.00	0.81
Consist busting	85.0	49.4	85.0	68.23	85.0	70.74	85.0	62.79

Table 4a Sensitivity of the Solution to Light Travel

Consist busting: 50%	No light travel	Medium light travel	Sufficient light travel
Total number of locomotives used	1,268	1,210	1,192
Total cost ($)	9,431,228	9,242,092	9,182,314
Percentage active time (%)	39.14	44.44	45.26
Percentage deadheading time (%)	12.37	8.09	7.33
Percentage idling time (%)	48.49	46.67	46.06
Percentage light traveling time (%)	0.00	0.80	1.35

bound trains). In the ALS solution, the percentage of the time that locomotives are actively pulling trains is about 14% more than in the CSX solution. Therefore, our solution significantly increases locomotive productivity.

We also conducted some tests to measure the sensitivity of the solution to the extent of light travel and the number of train-train connections; for the sake of brevity, we will present these results for one Scenario 1 only. Table 4(a) displays the statistics of the final solution when we allow three levels of light travel: no light travel; medium light travel; and sufficient light travel. We observe that a reasonable level of light travel can increase the locomotive utilization rate significantly. Table 4(b) provides the statistics of the final solution when we consider several levels of train-train connections. We observe that consist-busting can be decreased at the expense of a lower locomotive-utilization rate and a requirement of more locomotives.

9. Summary and Conclusions

In this paper, we have presented the results of a study of a locomotive-scheduling problem faced by CSX Transportation, a major U.S. railroad company. We focused on the planning version of the locomotive-scheduling problem, for which we assign locomotive types to various trains. Our goal was to develop excellent plans along a number of dimensions. Our primary metric for evaluating a solution was to compare it to the existing software that the locomotive-planning division at CSX uses. This existing software is an in-house software that has been developed over a period of 10 years. While it has the substantial advantage of capturing many constraints and objectives not dealt with in the existing literature, it also has some significant limitations. In particular, it is unable to produce acceptable solutions without considerable manual adjustment of the solution produced. Our approach to solve the planning version of the locomotive-scheduling problem produced solutions that satisfied the constraints and business rules specified by CSX and offered considerable savings in cost, especially in terms of a significant reduction in the number of locomotives needed. Our model is the first approach to account for consist busting, light travel, and consistency of the solution in a unified framework. Our approach relies on the technique of separating the locomotive-scheduling problem into a two-step process; first we solve the daily locomotive-scheduling problem, and then we solve the weekly locomotive-scheduling problem. This approach works reasonably well for those situations in which a large number of trains run most days of the week.

By incorporating more realistic constraints and costs in our models, we needed to rely on a multistage heuristic procedure to solve the model. In particular, we were unable to solve MIPs involving many fixed-charge constraints. We avoided this with a sequential heuristic procedure for determining the three sets of the decision variables (i) light travel, (ii) train-train connections, and (iii) active and deadhead locomotive flow variables sequentially instead of determining their values through an integrated single model. Nevertheless, our sequential procedure heavily exploited information from a sequence of LPs, and thus relied

Table 4b Sensitivity of the Solution to Consist Busting

	Very high consist busting	High consist busting	Medium consist busting	Low consist busting	Minimum consist busting
Consist busting rate (%)	66.39	59.48	49.42	39.09	31.06
Number of locomotive used	1,155	1,165	1,210	1,281	1,340
Total cost ($)	9,052,779	9,064,822	9,242,092	9,532,021	9,772,958
Percentage active time (%)	46.68	45.74	44.44	42.67	41.25
Percentage deadheading time (%)	6.28	6.79	8.09	9.99	9.57
Percentage idling and dwelling time (%)	46.23	46.68	46.67	46.35	48.04
Percentage light-traveling time (%)	0.81	0.79	0.80	0.99	1.14

on a systemwide view of the entire planning problem. When we determine light-travel arcs, we consider locomotive flow variables as part of the LPs solved also. Similarly, when we determine train-train connection variables, we again consider locomotive flow variables in the decision process. Therefore, we do achieve a certain level of integration in our sequential approach. Perhaps in the future we will be able to obtain a better integration of these different sets of variables.

Acknowledgments
The authors would like to express their sincere thanks to the referees for their insightful comments that led to significant improvements of the paper. They would additionally like to acknowledge the help of their students, Anurag Chandra, Ozlem Ergun, Anurag Mehta, and Mukundhan Krishnaswamy, who participated in this project and contributed in different capacities. The project was partly funded by CSX Transportation, and the authors gratefully appreciate the CSX Locomotive Division for giving them the research grant and access to their data. This research was also partly funded by the National Science Foundation Grants DMI-9900087, DMI-0085682, DMI-9820998, and the Office of Naval Research Grant ONR N00014-98-1-0317.

References

Aarts, E. H. L., J. K. Lenstra, eds. 1997. *Local Search in Combinatorial Optimization*. John Wiley & Sons, New York.

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Ahuja, R. K., J. B. Orlin, D. Sharma. 2001a. Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem. *Math. Programming* **91** 71–97.

Ahuja, R. K., J. B. Orlin, D. Sharma. 2001b. A very large-scale neighborhood search algorithm for the combined through-fleet assignment model. Submitted to *INFORMS J. Comput.*

Ahuja, R. K., O. Ergun, J. B. Orlin, A. P. Punnen. 2002. A survey of very large-scale neighborhood search techniques. *Discrete Appl. Math.* **123** 75–102.

Chih, K. C., M. A. Hornung, M. S. Rothenberg, A. L. Kornhauser. 1990. Implementation of a real time locomotive distribution system. T. K. S. Murthy, R. E. Rivier, G. F. List, J. Mikolaj, eds. *Computer Applications in Railway Planning and Management*. Computational Mechanics Publications, Southampton, UK, 39–49.

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

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

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

Forbes, M. A., J. N. Holt, A. M. Watts. 1991. Exact solution of locomotive scheduling problems. *J. Oper. Res. Soc.* **42** 825–831.

Garey, M. S., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Glover, F., M. Laguna. 1997. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA.

Nou, A., J. Desrosiers, F. Soumis. 1997. Weekly locomotive scheduling at Swedish state railways. Technical Report G-97-35, GERAD, École des Hautes Etudes Commercials de Montréal, Canada.

Smith, S., Y. Sheffi. 1988. Locomotive scheduling under uncertain demand. *Transportation Res. Record* **1251** 45–53.

Ziarati, K., F. Soumis, J. Desrosiers, M. M. Solomon. 1999. A branch-first, cut-second approach for locomotive assignment. *Management Sci.* **45** 1156–1168.

Ziarati, K., F. Soumis, J. Desrosiers, S. Gelinas, A. Saintonge. 1997. Locomotive assignment with heterogeneous consists at CN North America. *Eur. J. Oper. Res.* **97** 281–292.