

Solving Real-Life Locomotive Scheduling Problems

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

Jian Liu
Department of Industrial and Systems Engineering
University of Florida, Gainesville, FL 32611, USA
liujian@ufl.edu

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

Dushyant Sharma
Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
dushyant@mit.edu

Larry A. Shughart
CSX Transportation
500 Water St. J-230
Jacksonville, FL 32202, USA
Larry_Shughart@csx.com

(April 11, 2002)

Solving Real-Life Locomotive Scheduling Problems

Ravindra K. Ahuja¹, Jian Liu², James B. Orlin³, Dushyant Sharma⁴, and Larry Shughart⁵

Abstract

The locomotive scheduling problem (or the locomotive assignment problem) is to assign a *consist* (a set of locomotives) to each train in a pre-planned train schedule so as to provide them sufficient power to pull them from their origins to their destinations. Locomotive scheduling problems are among the most important problems in railroad scheduling. In this paper, we report the results of a study of the locomotive scheduling problem faced by CSX Transportation, a major US railroad company. We consider the planning version of the locomotive scheduling model (LSM), where there are multiple types of locomotives and we need to decide the set of locomotives to be assigned to each train. We present an integrated model that determines the set of active and deadheaded locomotives for each train, light traveling locomotives from power sources to power sinks, and train-to-train connections (specifying which inbound train 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 the set of locomotives coming on an inbound train is broken into subsets to be reassigned to two or more outbound trains. A solution is said to be *consistent* over a week with respect to a train, if the train gets the same locomotive assignment each day it runs. We give a mixed integer programming (MIP) formulation of the problem that contains about 197 thousand integer variables and 67 thousand constraints. 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 developed a solution technique to solve this problem within 30 minutes of computation time on a Pentium III computer. When we compared our solution with the solution obtained by the software in-house developed by CSX, we obtained a savings of over 400 locomotives, which translates into savings of over one hundred million dollars annually.

¹ Ravindra K. Ahuja, Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA.

² Jian Liu, Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA.

³ James B. Orlin, MIT Sloan School of Management, MIT, Cambridge, MA 02139, USA.

⁴ Dushyant Sharma, Operations Research Center, MIT, Cambridge, MA 02139, USA.

⁵ Larry Shughart, CSX Transportation, Jacksonville, FL 32202, USA.

1. INTRODUCTION

Transportation is one of the most vital services in modern society. Transportation of goods by railroads is an integral part of the US economy. Railroads play a leading role in multi-modal and container transportation. The rail transportation industry is very rich in terms of problems that can be modeled and solved using mathematical optimization techniques. However, research in railroad scheduling has experienced a slow growth and, until recently, most contributions used simplified models or had small instances failing to incorporate the characteristics of real-life applications. The strong competition facing rail carriers (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 growing interest for optimization techniques in railroad problems. In the last few years, a growing body of advances concerning several aspects of rail freight and passenger transportation has appeared in the operations research literature (see, for example, Jovanovic and Harker [1991], Brannlund et al. [1998], Newton et al. [1998], Cordeau et al. [1998], Sherali and Suharko [1998]). This paper concerns the development of new models and algorithms for solving real-life locomotive scheduling problems faced by US railroad companies.

The locomotive scheduling problem (or the locomotive assignment problem) is to assign a *consist* (a set of locomotives) to each train in a pre-planned train schedule so as to provide them sufficient power to pull them from their origins to their destinations. Locomotive scheduling problems are among the most important problems in railroad scheduling (Florian et al. [1976], Mao and Martland [1981], Smith and Sheffi [1988], Chih et al. [1990], Forbes et al. [1991], Fischetti and Toth [1997], Nou et al. [1997], and Ziarati et al. [1997, 1999]). Often, locomotive availability is the constraining factor in 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 types of locomotives, different types of trains, and the diverse geographic networks create very difficult combinatorial optimization problems. There are not satisfactory algorithms to solve these problems. As a result, there are inefficiencies in locomotive management, and substantial savings may be achieved through the development of better models and algorithms. CSX Transportation has over 3,000 locomotives, which translates into a capital investment of over \$5 billion, and over \$1 billion in yearly maintenance and operational costs. Our study reports an improvement of 5% in average locomotive utilization for CSX, which translates into a saving of over \$100 million per year.

Locomotive scheduling problems can be studied at two levels - planning level or operational level. At the planning stage of the locomotive scheduling problem, we assign locomotive types to various trains. Typically, a railroad company has different type of locomotives with different pulling and cost characteristics. For example, we may assign two CW44AC and one CW40-8 locomotive to a train. But the railroad company may have several hundred locomotives of type CW44AC and CW40-8. Of these, which specific units get assigned to the train is handled by the operational locomotive scheduling problem. The operational locomotive scheduling model also takes into account the fueling and maintenance needs of the locomotives, which are ignored in the planning model.

Railroad companies differ on their views of the relative importance of the planning problem versus the operational problem. One view holds that the day-to-day variability in traffic patterns, locomotive unreliability, changing service priorities and the wide range of train schedule operations create an environment that is so unpredictable that it is a futile exercise to develop a static locomotive scheduling plan. This philosophy, which we refer to as the tactical planning philosophy, would attempt to take into account current conditions on an on-going basis and dynamically solve the locomotive scheduling problem. This creates an ever changing set of solutions where particular trains may run with very

different consists each day, and each train may source those locomotives from a variety of inbound trains, depending on that days' situation.

These philosophy differences carry over to many aspects of the business. Ultimately, they can be summarized in two camps: those who believe it is best to operate a scheduled railroad and those who believe it is best to operate a "tonnage" or tactical railroad. The first camp would hold that while running a specific schedule the same way every time may in fact result in local inefficiencies (such as trains with very few cars or terminals that may be over staffed on some shifts), the macro level total cost of the operation is minimized. The second camp would hold that the total cost can be reduced by tactically adjusting the operation to reduce the number of local inefficiencies. These strategic approaches are linked to the view of the role of line managers as well. The operational model based railroad requires managers to be flexible and adjust their local operations to an ever changing pattern of trains, cars, locomotives and crews. The local managers also are expected to make quick, tactical decisions to adjust the operations to reduce inefficiencies both at the local and the network level. The planning model based railroad assumes that managers will anchor their local operation to the foundation of regular, routine, repeatable network operations. Over time, they will make finite adjustments to their many sub-processes that optimize their responsibility area. This fine tuning of the operation is not possible in a tactical operation due to the lack of stability.

These strategic approaches are linked to the view of the role of line managers as well. The operational model based railroad requires managers to be flexible and adjust their local operations to an ever changing pattern of trains, cars, locomotives and crews. The local managers also are expected to make quick, tactical decisions to adjust the operations to reduce inefficiencies both at the local and the network level. The planning model based railroad assumes that managers will anchor their local operation to the foundation of regular, routine, repeatable network operations. Over time, they will make small adjustments to their many sub-processes that optimize their responsibility area. This fine tuning of the operation is not possible in a tactical operation due to the lack of stability.

The CSX managers who sponsored this research chose to emphasize the planning part of the locomotive assignment problem for several reasons. First of all, they believe that the planning problem is of value in running the railroad. The CSX management philosophy is to operate a scheduled railroad and believes the inherent value of a repeatable, routine, scheduled set of decisions will ultimately not only minimize total locomotive costs but also total operating costs while improving the service product. Second, the planning system could be used as part of a network planning tool to evaluate which engines to buy in the future, and to study the impact of modifying their schedule. Third, any technology developed for the planning problem could possibly be extended to deal with tactical decisions in an operational setting.

In this paper, we consider the planning version of the locomotive scheduling model (LSM). We will now summarize the features of the LSM to give the reader a better understanding of the problem. A large railroad company has a train schedule which consists of several hundred trains with different weekly frequencies. Some trains run each 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 hauls and take several days to go from their origins to their destinations. To power these trains, CSX has several thousand locomotives of different types. Some locomotives are AC powered, some DC powered; they have different manufacturers, and some are more powerful than others. In LSM, we assign a set of locomotives to each train in the weekly train schedule so that each train gets sufficient tractive effort (that is, gets sufficient pulling power) and sufficient horsepower (that is, gets sufficient speed), and the assignment can be repeated indefinitely week to 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 be *deadheaded* on trains. Deadheaded locomotives do not pull the train; they are just pulled by active locomotives from one place to another place. Deadheading plays an important role in locomotive scheduling models since it allows extra locomotives to be moved from the places where they are in surplus to the places where there 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 of locomotives since 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 as there are no cars attached.

Since 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 its consist goes to the pool of locomotives where new consists are formed. In the former 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 latter case we say that consist-busting takes place. Consist-busting leads to mixing of locomotives from inbound trains and regrouping them to make new consists. This is undesirable from several angles. First, consist-busting requires additional locomotive time 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, which potentially propagates 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. A major contribution of this paper is to explicitly model the economic impacts of consist-busting, and to reduce its impacts on the system. Moreover, a schedule that performs a lot of consist-busting may be too complex to be implementable in practice.

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. Another contribution of our paper is that we model the effects of inconsistency, and try to reduce the inconsistency in a schedule.

This paper reports the development of a locomotive scheduling model that models the assignment of active and deadheaded locomotives to trains, light traveling of locomotives, consistency and consist-busting decisions in an integrated model. The objective in the model is to minimize the total cost, which is the sum of the active locomotive costs, deadheading costs, light travel costs, consist-busting costs, locomotive usage costs, and the penalty for using single locomotive consists. The solution is required to provide sufficient power to every train in a timely fashion to meet their prescribed schedules. We first describe a mixed integer programming formulation. Unfortunately, this MIP formulation is too large to be solved to optimality or near-optimality. We solve this model heuristically using a combination of techniques taken from linear programming, integer programming, and neighborhood search algorithms.

Locomotive scheduling problems are similar to the airline scheduling problems where one assigns planes of different types to flight legs. The planning version of the locomotive scheduling problem studied in this paper is similar to the well known fleet assignment model (see, for example, Subramaniam et al. [1994], Hane et al. [1995]), and the operational version of the locomotive scheduling problem is similar to the aircraft routing problem (see, for example, Barnhart et al. [1998]). However, the locomotive

scheduling problem considered in this paper is substantially more difficult than the fleet assignment model (FAM). In the FAM, we assign a single plane to a flight leg, but in LSM we assign a set of locomotives to a train. In FAM, there is no deadheading, no light travel, and no consist-busting. Further, the popular and well-solved FAM models assume that the flight leg schedule is the same each day of the week; that is, they solve the daily scheduling problem and repeat it each day of the week. But in the locomotive scheduling problem, we need to consider the weekly schedule of trains, and train schedules do not repeat on a daily basis. Hence, the LSM is combinatorially much harder to solve than the FAM.

The locomotive division at CSX Transportation has developed in-house software for the LSM which was developed, refined, and improved over a period of 10 years. This software has two main parts: *consist-builder* and *locomotive scheduler*. The consist-builder assigns active consists to trains; that is, which set of locomotives will actively pull the trains. This assignment is done based on train types, geography and additional business rules. The locomotive scheduler then routes locomotives in the weekly train network so that each train gets the desired active consist. In order to provide the desired active consists to trains, deadheading of the locomotives is often necessary. The locomotive scheduler considers each locomotive type one by one and solves a minimum cost flow problem to determine locomotive flow in the network. Breaking the LSM into two distinct parts leads to inefficiencies. When consist-builder assigns active consists, it does not take into account how locomotives will flow in the network to provide those consists to trains. When locomotive flow is determined, then consists cannot be changed. Another source of inefficiency in their approach is that the locomotive scheduler solved the problem one locomotive type (or, one commodity) at a time. In addition, this approach did not handle light travel of locomotives (which was done manually) and was unable to model consist-bustings. The planning solution provided by their software had very high consist-bustings. As many as 85% of the trains had their consists busted. Our model integrates both parts of their system into a single model, treats all locomotive types simultaneously, and models the impacts of consist-bustings, light travel, and deadheading.

Our LSM model is substantially different than locomotive scheduling models studied previously by researchers. Single locomotive models have been studied by Forbes et al. [1991], and Fischetti and Toth [1997]. Multicommodity flow based models for planning decisions have been studied by Florian et al. [1976], Smith and Sheffi [1988], Nou et al. [1997]. Multicommodity flow based models for operational decisions have been developed by Chih et al. [1990], and Ziarati et al. [1997, 1999]. Our multicommodity flow based model for planning decision has 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. The underlying flow network is the weekly space-time network where arcs denote trains, nodes denote events (that is, arrival and departure of trains), and different locomotive types define different commodities. Since we assign only integer number of locomotives to trains, we get integer multicommodity flow problems. The constraints that the locomotives assigned to a train must provide sufficient tonnage and horsepower and that the number of locomotives of each type is in limited quantity gives rise to the side constraints. In addition, our formulation has fixed charge variables which result from modeling the light travel and consist-bustings. Even when we ignore the consistency constraints, our formulation contains about 197 thousand integer variables and 67 thousand constraints, and is too large to be solved to optimality or near-optimality using existing commercial-level MIP software.

We developed a methodology to solve the locomotive scheduling problem heuristically. By using linear programming, mixed integer programming, and very large-scale neighborhood search techniques, we have attempted to obtain very good solutions of the locomotive scheduling problem. We solve the LSM in two stages. In the first stage, we modify 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 modify the solution of the first stage to solve the original problem where trains do not run all seven days a week. The first stage problem, though substantially smaller than the original problem, is still too large to be solved by the existing MIP software. The source of difficulty is the fixed charge variables introduced by modeling the light travel and consist-bustings. We developed linear programming based greedy algorithms to determine the values of these variables. Once these variables were determined, the resulting MIP gave very good solutions quickly, usually within 10 minutes.

We developed prototype software for our algorithms and compared our software with the in-house software used by CSX. On one benchmark 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 on 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.

This paper is organized as follows. In Section 2, we describe the problem in greater detail and define our notation. In Section 3, we describe the space-time network which will be the basis of all of our formulations. Section 4 describes the MIP formulation of the problem. In Section 5, we show the motivation for solving the problem in two stages, first as a daily scheduling problem followed by the weekly scheduling problem. Section 6 describes how we solve the daily scheduling problem, and Section 7 the weekly scheduling problem. In Section 8, we present a summary of our algorithmic approach. Section 9 presents the computational results of our approach and compares with the approach used at the CSX Transportation. Finally, Section 10 summarizes our contributions and outlines the future research issues.

2. PROBLEM DETAILS

In this section, we give the details and notation of the locomotive scheduling problem used for planning at CSX.

Train Data:

Locomotives pull a set L of trains from their origins to destinations. The train schedule is assumed to repeat from week to week. 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 for which all data is the same except that they will have different departure and arrival times.

We use the index l to denote a specific train. For the planning model, the train schedule is deterministic and pre-specified. There are three classes of trains: *Auto*, *Merchandize*, and *Intermodal*. Each train belongs to exactly one class. The required tonnage and horsepower is specified. The tonnage of a train represents the minimum pulling power needed to pull the train. The tonnage depends upon the number of cars pulled by the train, weight of the cars, and the slope or ruling grade of that train's route. The horsepower required by the train is its tonnage multiplied by the factor that we call the *horsepower per tonnage*. The greater the horsepower per tonnage, the faster the train can move. Different classes of trains have different horsepower per tonnage. For greater model flexibility, we allow each train to have its own horsepower per tonnage. We associate the following data with each train l .

$dep-time(l)$: The departure time for the train l . We express this time in terms of the weekly time as the number of minutes past Sunday midnight. For example, if the train l leaves on Monday 6 AM, then $dep-time(l) = 360$; and if it leaves on Tuesday 6 AM, then $dep-time(l) = 1,800$.

$arr-time(l)$: The arrival time for train l (in the same format as the $dep-time(l)$).

$dep-station(l)$: The departure station for train l .

$arr-station(l)$: The arrival station for train l .

T_l : Tonnage requirement of train l .

β_l : Horsepower per tonnage for train l .

H_l : Horsepower requirement of train l , which is defined as $H_l = \beta_l T_l$.

E_l : The penalty for using a single locomotive consist for train l .

Locomotive Data:

A railroad company typically has several different types of locomotives with different pulling and cost characteristics and different number of axles (often varying from 4 to 9). Locomotives with different characteristics allow railroads greater flexibility in locomotive assignments, but also make the locomotive scheduling problem substantially more difficult. We denote by K the set of all locomotive types, and use the index k to represent a particular locomotive type. We associate the following data with each train $k \in K$:

h^k : Horsepower provided by a locomotive of type k .

α^k : Number of axles in a locomotive of type k .

G^k : Weekly ownership cost for a locomotive of type k .

B^k : Fleet size of locomotives of type k , that is, the number of locomotives available for assignment.

Active and Deadheaded Locomotives:

Locomotives assigned to a train either actively pull the train or deadhead. Deadheading allows extra locomotives to be moved from places where they are in surplus to the places where they are in short supply. For example, more tonnage leaves a coal mine than arrives at the coal mine; so more pulling power is needed on trains departing from the mine. This creates a demand for locomotives at the mine. Similarly, more tonnage arrives at a thermal power plant than leaves it; so more pulling power is needed on trains arriving at the power plant. This creates a surplus of locomotives at the power plant. Effective deadheading of locomotives reduces the total number of locomotives used and improves the average locomotive utilization.

We need the following data for train-locomotive type combinations:

c_l^k : The cost incurred in assigning an active locomotive of type k to train l .

d_l^k : The cost incurred in assigning a deadheaded locomotive of type k to train l .

t_l^k : The tonnage pulling capability provided by an active locomotive of type k to train l .

The active cost c_l^k captures the economic asset cost of the locomotive for the duration of the train and the fuel and maintenance costs. The deadhead cost d_l^k captures the same asset cost, a reduced maintenance cost, and zero fuel cost. Observe that the tonnage provided by a locomotive depends upon the train. Different train routes have different ruling grades (that is, slopes) and the pulling power provided by a locomotive type is affected by the ruling grade.

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. CSX uses business rules based on train types and geographical considerations to determine these classes for each train. 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]*).

Light Travel:

Our model allows light travel of locomotives, that is, locomotives traveling in a group on their own between different stations to reposition themselves. Similar to deadheading, light travel can be an effective way to reposition locomotives. The light travel cost has a fixed component that depends upon the distance of travel in the light move since we need a crew, and a variable component that depends upon the number of locomotives light traveling.

Consist-Busting:

Consist-busting is a normal phenomenon in railroads because the needs for outgoing locomotives at a station do not precisely match the incoming needs. However, consist-busting incurs a cost in complexity of managing the system, and in delays in repositioning locomotives. Consist-busting can be reduced by a better scheduling of locomotives. We model the cost of consist-busting with a fixed component, B , per consist-busting and a variable component that depends upon the number of locomotives involved in the consist-busting.

We will now describe the constraints in the LSM. The constraints can be classified into two parts: *hard constraints* (which each locomotive assignment must satisfy) and *soft constraints* (which are desirable but not always required to be satisfied). We incorporate soft constraints by attaching a penalty for each violation of these constraints.

Hard Constraints:

Power requirement of trains: Each train must be assigned locomotives with at least the required tonnage and horsepower.

Locomotive type constraints: Each train l is assigned locomotive types belonging to the set *MostPreferred[l]* and *LessPreferred[l]* only.

Locomotive balance constraints: The number of incoming locomotives of each type into a station at a given time must equal the number of outgoing locomotives of that type at that station at that time.

Active axles constraints: Each train must be assigned locomotives with at most 24 active axles. This business rule is designed to protect the standard couplers used in North America. Exceeding 24 powered axles may result in overstressing the couplers and causing a train separation.

Consist size constraints: Each train can be assigned at most 12 locomotives including both the active and deadheaded locomotives. This rule is a business policy of CSX that reduces its risk exposure if the train were to suffer a catastrophic derailment.

Fleet size constraints: The number of assigned locomotives of each type is at most the number of available locomotives of that type.

Repeatability of the schedule: The number and type of locomotives positioned at each station at the beginning of the week must equal the number and type of locomotives positioned at that station at the end of the week. This ensures that the assignment of locomotives can be repeated from week to week.

Soft Constraints:

Consistency in locomotive assignment: If a train runs five days a week, then it should be assigned the same consist each day it runs. CSX believes that crews will perform more efficiently and more safely if they operate the same equipment on a particular route and train. As the crews learn the operating nuances associated with each combination, they will adjust their throttle and braking control accordingly.

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 on each day both the trains run. This is useful to help terminal managers optimize their sub-processes associated with arriving and departing trains.

Same class connections: Trains should connect to other trains in the same class, e.g., auto trains should connect to auto trains; merchandise trains should connect to merchandise trains, etc. This is useful in that different trains have different preferred types of locomotives and may originate and terminate at different locations within a larger terminal area (an unloading ramp, for example).

Avoid consist-busting: Consist-busting should be avoided as much as possible.

OBJECTIVE FUNCTION:

The objective function for the locomotive scheduling model contains the following terms:

- Cost of ownership, maintenance, and fueling of locomotives
- Cost of active and deadheaded locomotives
- Cost of light traveling locomotives
- Penalty for consist-busting
- Penalty for inconsistency in locomotive assignment and train-to-train connections
- 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 space-time network as $G^7 = (N^7, A^7)$, where N^7 denotes the node set and A^7 denotes the arc set. We construct the weekly space-time network as follows. We create a

train arc (l, l'') for each train l ; the tail node l' of the arc denotes the event for the departure of train l at $\text{dep-station}(l)$ and is called a *departure node*. The head node l'' denotes the arrival event of train l at $\text{arr-station}(l)$ and is called an *arrival node*. Each arrival or departure node has two attributes: place and time. For example, $\text{place}(l') = \text{dep-station}(l)$ and $\text{time}(l') = \text{dep-time}(l)$. Similarly, $\text{place}(l'') = \text{arr-station}(l)$ and $\text{time}(l'') = \text{arr-time}(l)$. Some trains are called *forward trains* and some trains are called *backward trains*. *Forward trains* are those trains for which $\text{dep-time}(l) > \text{arr-time}(l)$ and *backward trains* are those trains for which $\text{dep-time}(l) < \text{arr-time}(l)$. For example, a train that leaves on Monday and arrives at its destination on Tuesday is a forward train; whereas a train that leaves on Saturday and arrives on Monday is a backward train. (Recall that our timeline begins on Sunday at midnight.)

To allow the flow of locomotives from an inbound train to an outbound train, we introduce *ground nodes* and *connection arcs*. For each arrival node, we create a corresponding *arrival-ground node* with the same place and time attribute as that of the arrival event. Similarly, for each departure event, we create a *departure-ground node* with the same place and time attribute as that of the departure node. 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 the chronological order of their time attributes, and connect each ground node to the next ground node in this order 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 nodes at a station represent the pool (or storage) of locomotives at the station at different instants of times, when events take place. As trains arrive, they bring in locomotives to the pool through arrival-ground connection arcs. As train departs, they take out locomotives from the pool through ground-departure connection arcs. The ground arcs allow inbound locomotives to stay in the 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 becoming 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 those departure nodes whenever such a connection can be feasibly made. Railroads have some business rules about which train-train connections can be feasibly made. The arrival nodes i'' for a train i can be connected to a departure node j' for train j provided $\text{min-connection-time} \leq \text{dep-time}(j) - \text{arr-time}(i) \leq \text{max-connection-time}$, where *min-connection-time* and *max-connection-time* are two specified parameters. For example, $\text{min-connection-time} = 120$ (in minutes) and $\text{max-connection-time} = 480$.

We also allow the possibility of light travel, that is, several locomotives forming a group and traveling on their own as a group from one station to another station. Using a method described later in Section 6.4, we create possibilities for light travel among different stations. We create a *light arc* in the weekly space-time network corresponding to each light travel possibility. 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 which 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 which depends upon the number of locomotives light traveling as a group.

To summarize, the weekly space-time network $G = (N, A)$ has three types of nodes - arrival nodes (*ArrNodes*), departure nodes (*DepNodes*), and ground nodes (*GrNodes*); and four kinds of arcs - train arcs (*TrArcs*), connection arcs (*CoArcs*), ground arcs (*GrArcs*) and light travel arcs (*LiArcs*). Let $\text{AllNodes} = \text{ArrNodes} \cup \text{DepNodes} \cup \text{GrNodes}$, and $\text{AllArcs} = \text{TrArcs} \cup \text{CoArcs} \cup \text{GrArcs} \cup \text{LiArcs}$. We show in

Figure 1, a part of the weekly space-time network at a particular station, which illustrates various kinds of arcs.

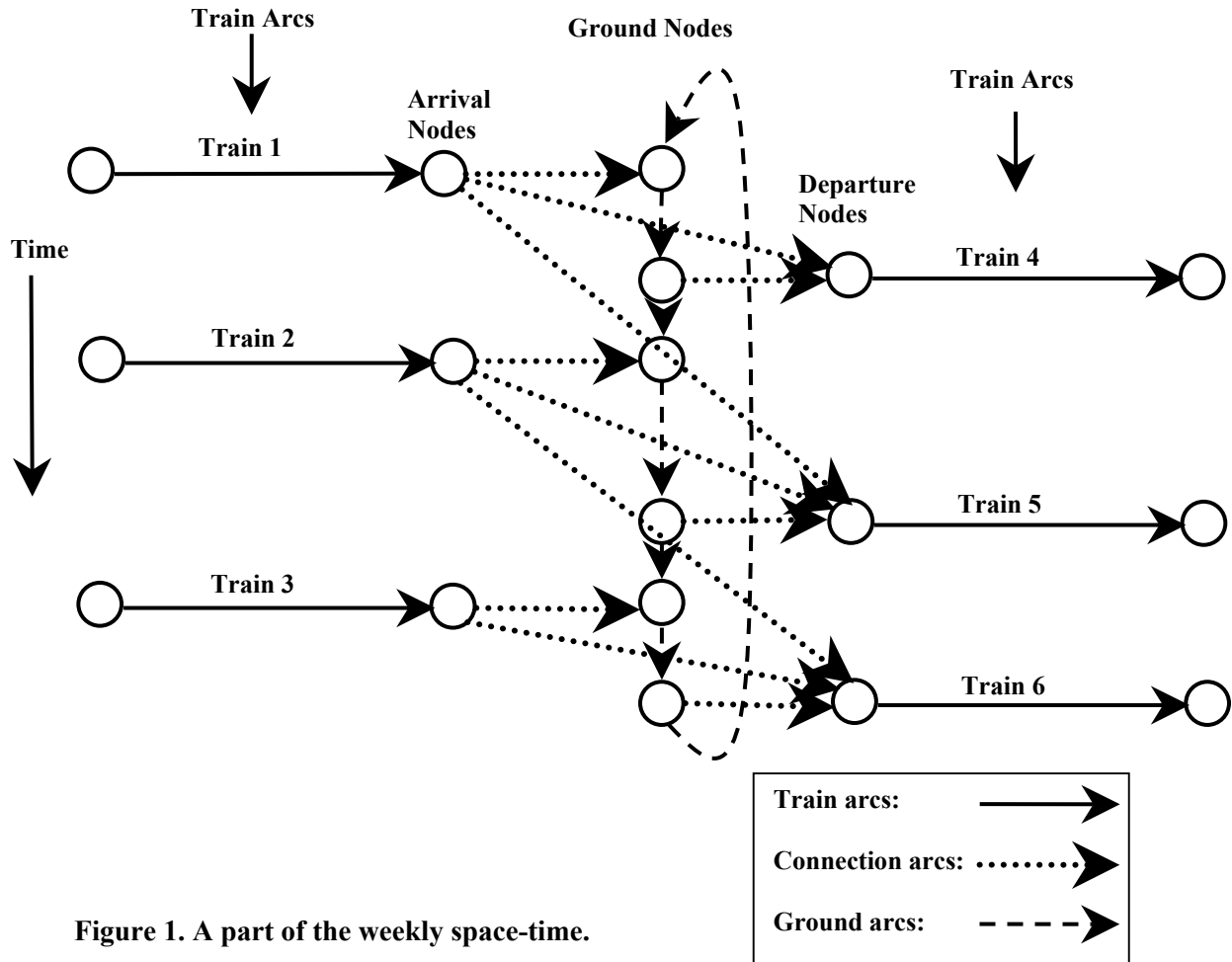


Figure 1. A part of the weekly space-time.

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 either are *active* or *deadheading*, those flowing on light arcs are *light traveling*, and those flowing on connection 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:

$I[i]$: Set of incoming arcs into node $i \in AllNodes$.

$O[i]$: Set of arcs emanating from node $i \in AllNodes$.

$tail(l)$: The tail node of arc $l \in AllArcs$.

$head(l)$: The head node of arc $l \in AllArcs$.

F_l : This cost term denotes the cost of an (active) light traveling locomotive with crew on a light arc $l \in LiArcs$. This cost includes the crew cost as well. If several locomotives light travel on an arc, then only one locomotive is active while others deadhead. For simplicity, we assume that the cost term is independent of the locomotive type pulling the consist.

- d_l^k : Recall that in Section 2, we defined d_l^k as the cost of deadheading of locomotive type k on train arc l . We now define it for every arc $l \in AllArcs$. For a light arc $l \in LiArcs$, d_l^k captures the cost of traveling for a non-active locomotive of locomotive type k on arc l . For a connection arc $l \in CoArcs \cup GrArcs$, d_l^k captures the cost of idling for locomotive type k on arc l .
- CB : The set of all connection arcs from arrival nodes to ground nodes. These are the arcs on which positive flow represents consist-busting. Alternatively, $CB = \{(i, j) \in AllArcs: i \in ArrNodes \text{ and } j \in GrNodes\}$.
- $CheckTime$: It is a time instant of the week when no event takes place; that is, no train arrives or departs at any station. We will henceforth assume that $CheckTime$ is Sunday midnight.
- S : The set of arcs that cross the $CheckTime$; that is, $S = \{(i, j) \in AllArcs: time(i) < CheckTime < time(j)\}$.

4. THE MIXED INTEGER PROGRAMMING FORMULATION

In this section, we present the mixed integer programming (MIP) formulation of the locomotive scheduling model.

Decision Variables:

x_l^k : Integer variable representing the number of active locomotives of type $k \in K$ on the arc $l \in TrArcs$;

y_l^k : Integer variable indicating the number of non-active locomotives (deadheading, light-traveling or idling) of type $k \in K$ on the arc $l \in AllArcs$;

z_l : Binary variable which takes value 1 if at least one locomotive flows on the arc $l \in LiArcs \cup CoArcs$, and 0 otherwise;

w_l : Binary variable which takes value 1 if there is a flow of a single locomotive on arc $l \in TrArcs$, and 0 otherwise;

s^k : Integer variable indicating the number of unused locomotives of type $k \in K$.

Objective Function:

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

Constraints:

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

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

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

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

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

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

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

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

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

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

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

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

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

We now give some explanation for the above formulation to show that it correctly represents the LSM. We first discuss the constraints. The constraint (1b) ensures that the locomotives assigned to a train provide the required tonnage, and the constraint (1c) ensures that the locomotives assigned provide the required horsepower. The 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 is assigned at most 12 locomotives. The flow balance constraints (1f) ensures that the number of incoming locomotives equal the number of outgoing locomotives at every node of the weekly space-time network. The 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. The 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). The 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. The constraint (1j) makes the variable w_l equal to 1 whenever a single locomotive consist is assigned to train l . Finally, the constraint (1k) counts the total number of locomotives used in the week; which is the sum of the flow of locomotives on all the arcs crossing the *CheckTime*. The difference between the number of locomotives available minus the number of locomotives used gives the number of locomotives saved (s^k).

We 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 and light travel arcs, and 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, $\sum_{l \in CB} B z_l$, 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 the formulation (1) assumes that any locomotive type can flow on any train arc. But recall from our discussion in Section 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 the formulation (1) we add the constraints $x_l^k = 0$ for each $k \in \text{Prohibited}[l]$ and each $l \in \text{TrArcs}$. This constraint ensures 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 (for example, 1.2) for each $k \in \text{LessPreferred}[l]$ and each $l \in \text{TrArcs}$.

Our formulation (1) incorporates all the hard constraints described in Section 2, but not all the soft constraints. The formulation does not incorporate the consistency constraints. Including those constraints would make the formulation unmanageably large. We, however, do handle the same class constraints implicitly in the definition of the term d_l^k . If a train-train connection arc l is not the same class connection arc, we multiply its d_l^k value by a constant parameter greater than 1, which penalizes the use of these arcs. By suitably selecting the value of this parameter, we can satisfy the same class constraints to varying degrees. We can also obtain different levels of consist-busting by selecting different values of the consist-busting cost B . The greater the value of B we choose, the less will be the amount of consist-busting in an optimal solution.

In the data provided to us by CSX, there were 538 trains, each of which operate several days in a week, and 5 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 state of the art commercial software. As a matter of fact, we were unable to solve even the linear programming (LP) relaxation of the problem. When we solved the LP relaxation of a much smaller problem, we found that the LP solution was highly fractional. Most of the variables were non-integer and the integer programming algorithm did not find a good integer solution in several hours. In addition, our MIP formulation (1) omits the consistency constraints that the same train running on different days should have the same locomotive assignment. In our formulation, a train running on different days is treated as separate trains and their locomotive assignments can be quite different. In principle, we can add new constraints to enforce the consistency constraints, but that would be impractical as it would increase the size of the MIP substantially. Hence, satisfying the consistency constraints by introducing new constraints did not seem to be a feasible option. Rather we developed an alternative approach that simultaneously helped enforce consistency while dramatically reducing the problem size. We describe our approach in the next section.

5. SIMPLIFYING THE MODEL

We next analyzed the number of trains running with different frequencies in a week for CSX. The table shown in Figure 2 gives these numbers. The first column in the table gives the train frequency in a week; that is, how often the trains run in a week. The second column in the table gives the number of trains in the train schedule with the frequency given in the first column. For example, in the train schedule, there are 372 trains that run all seven days a week, there are 62 trains that run six days a week, and so on. The third column is a product of the first and second columns, the fourth column gives a cumulative sum of the third column, and the fifth column expresses the values in the fourth column in the percentage notation. Observe that the third column gives the number of train arcs in the weekly space-time network for trains with different frequencies. The table shows that 94% of the train arcs in the space-time network correspond to the trains that run 5, 6, or 7 days.

Train Frequency (A)	Number of Trains (B)	A*B	Cum. Sum of A*B	Cumulative Percentage A*B
7	372	2,604	2,604	78%
6	62	372	2,976	90%
5	29	145	3,121	94%
4	24	96	3,217	97%
3	20	60	3,277	99%
2	16	32	3,309	100%
1	15	15	3,324	100%

Figure 2. Analysis of trains and their frequencies.

We now state 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. Each train with a frequency of p days per week or larger is a train in the daily model. Each train with frequency strictly less than p days is not included in the daily model. After some analysis and testing, we chose $p = 5$. So, if a train has a frequency of 4 or less in the weekly model, we eliminate it from our daily model. Our daily model is roughly seven times smaller than the weekly model, and accordingly much easier to solve. In addition, since we repeat the train's daily schedule for each day of the week, the solution automatically satisfies the consistency constraints. Hence this simplification achieves both of our goals, which is to reduce the problem size and achieve consistency of the solution.

The solution to the daily model is not feasible for the weekly model. It assigns locomotives to some train arcs that do not exist (those trains that run 5 or 6 days per week) and does not assign locomotives to train arcs that exist (those trains that run 4 days per week or fewer). But for the CSX data, the approximation we make is relatively small compared to the size of the problem. With our choice of $p = 5$, we assign locomotives to 120 train arcs that do not exist in the weekly model and do not assign locomotives to 203 train arcs that do exist in the weekly model. These numbers are small compared to the total number of train arcs, which is 3,324.

We take the solution for the daily model and transform it to a feasible solution to the weekly model in a separate algorithm. We thus solve the locomotive scheduling problem in two stages. The first stage solves the daily locomotive scheduling problem, and the second stage takes in the daily locomotive schedule and modifies it to obtain the weekly locomotive schedule. We will in the next few sections focus on the daily locomotive scheduling problem, and return to the weekly locomotive scheduling problem in Section 7. In our investigations, we chose $p = 5$ primarily because the solution of the model with $p = 5$ was more easily converted to a solution of the weekly locomotive scheduling problem than other values of p .

6. SOLVING THE DAILY LOCOMOTIVE SCHEDULING PROBLEM

In this section, we describe how to solve the daily scheduling problem. We describe (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 where each train is assumed to run each day of the week. It is constructed in a similar manner to the way in

which we construct the weekly space-time network. We represent the daily space-time network as $G^l = (N^l, A^l)$. 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 will cross the midnight time line as it goes from its origin to its destination. For example, consider a train that leaves station A on 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 4PM on Monday, and the one that starts at 7 AM on Sunday and ends at 4 PM on Tuesday. In general, if $day(l) = k$ in our daily model, then there are k copies of the train l that cross the midnight at Sunday time line in our weekly model. Therefore, assigning a locomotive in our daily model to a train l with $day(l) = k$ corresponds to assigning k 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 given 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 give 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 take 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.

In order to obtain high quality feasible solutions and to keep the total running time of the algorithm small, 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 kind 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 connections arcs, followed by the fixed charge variables corresponding to light arcs. We 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 hard-wired train-train connections are easy to enforce. If the inbound train i at a station is hardwired to the outbound train j at that station, then in the space-time network we create the train-train connection arc (i'', j') and do not

allow any other connection arc to emanate node i'' or enter node j' . This arc is the only connection arc leaving node i'' and the only connection arc entering node j' . This will ensure that all locomotives brought in by the inbound train i directly go to the outbound train j without consist-busting. In this case, there will be no fixed charge variables corresponding to the trains i and j .

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, a station has one inbound train and two outbound trains, or it has two inbound trains and one outbound train. In these cases too, consist-busting cannot be avoided and there is no need to introduce any train-train connection arcs. Again, there will not be any fixed charge variables; all inbound train(s) send their locomotives to the ground node, and all outbound train(s) will get their locomotives from the ground node.

Train companies also have rules about which train-train connections are permissible or desirable. The difference between the arrival time of the inbound train and the departure time of the outbound train must be between some specified lower and upper bounds in order for a valid train-train connection to be made. They also prefer connections between trains of the same class; that is, auto trains sending locomotives to auto trains and merchandize trains to the merchandize trains, etc. They also want that the tonnage and HP requirements of the two trains be similar in order for it to be a desirable train-train connection. Using these business rules, we determine the set of all “candidate” train-train connections. In the next step, we will “fix” some of these candidate train-train connections into hardwired train-train connections and all “unfixed” trains will send (or receive) locomotives to (from) ground nodes. Using these heuristics we eliminate fixed charge variables corresponding to consist-busting in our model.

We fix some candidate train-train connections by solving an iterative procedure that solves a sequence of linear programming relaxations of the daily locomotive scheduling problem given in Section 6.1 with some changes. We start with a daily space-time network containing all the candidate train-train connection arcs and candidate light arcs. We next solve the linear programming relaxation of the locomotive scheduling model where 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. This is an alternative way of discouraging consist-busting. Let $\alpha(l)$ 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 $\alpha(h)$. This arc indicates a good potential train-train connection. We make this connection arc the unique connection arc for the two corresponding trains, that is, we make it mandatory for the inbound train to send its locomotive to the corresponding outbound train, and resolve the linear programming relaxation. If this linear programming relaxation is infeasible or increases the cost of the new solution by an amount greater than β , 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. This completes one iteration of our procedure to fix train-train connections. We select another candidate train-train connection arc h with the largest value of $\alpha(h)$ 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 the set of candidate train-train connections becomes empty.

Our method of identifying the train-train connections is a greedy method. It identifies a promising choice, makes it, and resolves the LP problem to assess the impact of this choice. If it turns out to be a bad choice, it is ignored; otherwise, it is made. By suitably identifying the values of the two parameters β and γ , we can govern the behavior of the algorithm. By choosing the higher value of β , 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 show in Section 9 that 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 an achievable goal. The parameters β and γ need to be assigned right values so that we get the desired consist-busting. We used $\beta = \$1,000$, 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 should be the light travel arcs in the space-time network; and (ii) *flow on light travel arcs*: which locomotives will 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 would need to add a large number of arcs in the daily space-time network. This would increase the size of the network substantially and also the number of flow variables. In addition, since we associate a fixed charge variable with each candidate light arc, the number of fixed charge variables would be very high. Rather than 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, and another heuristic procedure for selecting a subset of these arcs for light travel. We 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). We first construct the *space network*, which is the same as the daily space-time network described in Section 6.1 except that we ignore the time element. Equivalently, we shrink all the station nodes with different times into a single station node. We next determine the additional availability of pulling power at the power sources and the additional demand for pulling power at the 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 some standard type, such as, SD40. 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 city i to city j in the space-time network departing from city i every 8 hours. For the data provided to us by CSX, we found 58 arcs in the space network with flow greater than 2 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 these many fixed charge variables in the required time. Our second procedure eliminates the fixed charge variables. From the list generated by the previous heuristic, it selects a small subset of light arcs and assumes that there will be light travel on these arcs. The procedure works as follows. It first solves a linear programming formulation of the locomotive scheduling problem where all the candidate light arcs are added. It then removes all those light arcs which have zero flow. Let $\alpha(l)$ denote the total number of locomotives flowing on an arc l . We then perform the following iterative step. We select the light arc l with the smallest value of flow $\alpha(l)$. Suppose this is arc q . We delete arc q from the network and solve the LP relaxation of the locomotive scheduling problem. If

the deletion of this light arc causes the total cost of flow (of the LP relaxation) to go up by at least η units, we do not delete this arc; otherwise we delete this arc and replace α by the new flow. We repeat this process until there are no unexamined 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 will increase the number of arcs deleted from the network which will result in lesser number of light arcs being generated. Hence, by assigning suitable values to this parameter, one can generate 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:

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

subject to:

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

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

$$\sum_{k \in K} a^k x_l^k \leq 24, \quad \text{for all } l \in TrArcs, \quad (2d)$$

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

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

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

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

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

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

We refer to a solution of (2) by (x, y) , since w can be deduced using the solution (x, y) . The integer programming model here is much easier to solve than the model described in Section 6.1. It does not contain fixed charge variables and has fewer locomotive flow variables. For the locomotive scheduling problem we solved, it 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 gave a very good 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, did not improve much over the best integer solution obtained within 15 minutes; hence prematurely terminating the algorithm after 15 minutes did not affect much 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 of the locomotive scheduling problem. We found that this solution can be easily improved by a modest perturbation 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 which 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. We use the solution obtained by the integer programming software as the starting solution and improve it using a VLSN search algorithm.

The locomotive scheduling problem can be conceived of as a multicommodity flow problem with side constraints in the space-time network where the flow of each locomotive type defines a commodity. When viewed in this manner, the constraints (2f) define the flow balance constraints of each commodity and the remaining constraints define the side constraints. The multicommodity flow problem is polynomially solvable when the flow variables x_l^k and y_l^k take real values, but is NP-complete when flow variables are required to take integer values. (For the time being, we are ignoring the fixed penalty for single locomotive trains captured by the constraints (2g).)

We define the neighborhood for a feasible solution (\bar{x}, \bar{y}) of (2) as follows: For a specified value of $k \in K$, send $\delta > 0$ locomotives of type k along a cycle in the daily space-time network so that the flow balance and all side constraints remain satisfied. The new solution is a lower cost solution (or a better neighbor) if the cycle along which the flow is sent is a negative cost cycle. In this neighborhood, the neighborhood search algorithm would proceed by identifying negative cost cycles with respect to a given solution (\bar{x}, \bar{y}) and a locomotive type k and augmenting flows along these cycles until there is no negative cost cycle for any locomotive type k .

We now describe how we identify negative cost cycles for a given locomotive type k . Let α_l^k denote the total number of locomotives of type k on arc l (that is, $\alpha_l^k = x_l^k + y_l^k$). We first create a *residual network* $G(\alpha, k)$, similar to those used in solving minimum cost flow problems (see, for example, Ahuja, Magnanti and Orlin [1993]). We consider each arc $l \in AllArcs$ and add arcs to the residual network in the manner described below:

Case 1: If we can send additional locomotives of type k on arc l , then we add arc l to the residual network. The capacity u_l^k of the arc l is equal to the maximum number of locomotives of type k that can be sent on arc l without violating any of constraints in (2b)-(2e) and (2h)-(2j). The cost f_l^k of the arc l is equal to the increase in the cost of sending one additional locomotive on the arc l .

Case 2: If we can reduce locomotives of type k on arc l , then we add the reversal of arc l , say, arc \bar{l} , to the residual network. The capacity $u_{\bar{l}}^k$ of the arc \bar{l} is equal to the maximum number of locomotives of type k that can be reduced on arc l without violating any of constraints in (2b)-(2e) and (2h)-(2j). The cost $f_{\bar{l}}^k$ of the arc \bar{l} equals the decrease in the cost of sending one fewer locomotive of type k on arc l .

For any negative cost directed cycle W in the residual network, there is a way to improve in the current solution. If the cost of cycle W is c_W and if its capacity is δ units (the capacity of W is the

minimum capacity of an arc in W), then δ locomotives of type k can be sent around the cycle W resulting in a total savings of δc_w . There exist several efficient negative cycle detection algorithms (see, for example, Ahuja et al. [1993], and Goldberg et al. [1995]), and any of these algorithms can be used to identify negative cycles.

The neighborhood for a given solution (\bar{x}, \bar{y}) is defined so that each neighbor is any solution that can be obtained by sending one or more units of flow around one of the directed cycles in the residual network. The number of directed cycles in each residual network can be very large and may not be polynomially bounded in terms of the number of nodes and arcs in the network. Hence our neighborhood search algorithm is a very large-scale neighborhood (VLSN) search algorithm. In the VLSN search algorithms, the size of the neighborhood is too large to be searched explicitly and we use implicit enumeration methods to identify improved neighbors. Algorithms based on VLSNs have been successfully used in the context of optimization problems (Ahuja et al. [2001a, 2001b, 2001c, 2002]). To apply this algorithm, we start with a feasible solution of the locomotive scheduling problem, select a locomotive type k , construct the residual network, identify negative cycles in it, and send maximum possible flow along this cycle. We then update the residual network and again send flow along a negative cycle. We repeat this process until there is no negative cycle in the residual network. At this time, we select another locomotive type k' and reapply the method. We terminate when no negative cycle is found for any locomotive type.

This algorithm, when implemented, ran very efficiently. We were able to obtain a local optimal solution in a few seconds of computer time. The algorithm obtained some improvements over the solution obtained by CPLEX. Although our neighborhood was quite large, ultimately we decided that it was not large enough. The side constraints (2h) were often tight and did not permit many locomotives to be rerouted without violating the constraints. In order to find more substantial improvements, we need to permit flows to change in many arcs simultaneously. This observation motivated us to consider another neighborhood that we describe next. This neighborhood can be considered as a specific implementation of the concept of *referent domain optimization* briefly described in Glover and Laguna [1997].

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 $q \in K \setminus \{k\}$ 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}) .

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

subject to:

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

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

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

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

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

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

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

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

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

This new neighborhood subsumes our previous cycle-based neighborhood and is also a very large-scale neighborhood. 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. Though (3) is an integer programming problem with 2,168 variables and 3,437 constraints, CPLEX solved 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 locomotive 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 now describe how we solve the weekly locomotive scheduling problem. Recall from our discussion in Section 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 which 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 don't 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, and possibly use additional locomotives to meet the constraints. In this section, we describe this method in greater detail.

We define some notation to simplify the presentation. There are two kinds of trains in the locomotive scheduling problem: (i) Φ : trains that operate p or more days a week; and Ψ : 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 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, $\text{TrArcs} = \Phi^7 \cup \Psi^7$.

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 when we solve 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 be as close to its target flow as possible as this would ensure that the weekly locomotive schedule is consistent. Changes to the target flow can be made but they are penalized as it may affect the consistency of the solution. Henceforth, we will 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 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 10% of the train arcs are in Ψ^7 and ignoring consistency of these arcs will 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 the trains A and B run.

We now describe the weekly space-time network. The objective (iii) requires that we slightly modify the weekly space-time network defined earlier in Section 3. Recall that in the daily space-time network, we have train-train connection arcs which force the locomotives to go from an inbound train to an outbound train without consist-busting. When we create the weekly space-time network, we create these train-train connections on all days wherever it is possible. For example, suppose that the inbound train l_1 runs Monday through Friday, the outbound train l_2 runs Wednesday through Sunday, and we had a train-train connection from l_1 to l_2 in the daily space-time network. Then in the weekly space-time network, we will have train-train connection arcs only on Wednesday through Friday when both the trains run. On all other days, the train l_1 will send its locomotives to the ground node and the train l_2 will get its locomotive from the ground node. Those trains that do not have a train-train connection in the daily space-time network, send their locomotives to the corresponding ground node and get their locomotives from the corresponding ground node. Thus, each arrival node has exactly one outgoing arc; either it is a train-train connection arc or it is a train-ground connection arc. Similarly, each departure node has exactly one incoming arc. Either it is a train-train connection arc or it is a train-ground connection arc. The weekly space-time network also has light arcs. Each light travel arc in the daily space-time network is repeated seven times in the weekly space-time network to allow the possibility of light travel each day of the week.

We 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 Section 4. But 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. Since 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 these single commodity flow problems with side constraints were solved very efficiently by CPLEX 7.0. We now describe this approach in greater detail. We first arranged different locomotive types in some order, and then considered them one by one in this order.

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

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

subject to (3b)-(3g), (3i)-(3j), and the following constraints:

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

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

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

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

where M is a large positive number, and

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

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 the formulation (3), which is to find 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 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 the 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 arcs in the set Φ . In our computational investigations, we used $P = \$1,000$.

We solved the formulation (4) multiple times and with different objective functions to obtain improved solutions. For example, our primary objective is to provide sufficient number of locomotives to trains in Ψ . We accomplish it by choosing a sufficiently large value of M (say, 10^6), and solving (4) for all locomotive types. Once all trains have sufficient number of active locomotives, we set $M = 0$ and

solve it again for all locomotive types. Depending upon the value of P , the optimal solution will balance 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, the formulation (4) is defined over a network with 8,798 nodes and 9,139 arcs. The formulation (4) comprises 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 2-3 minutes to obtain a satisfactory solution in the weekly space-time network. Whereas the integer programming formulation of the weekly space-time network was intractable, 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 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 Section 6.6 for the daily space-time network.

Recall our discussion in Section 6.3 that we create train-train connections between several inbound and outbound trains to reduce consist-bustings. After we have obtained the solution of the weekly locomotive scheduling problem, we can create additional train-train connections. After we have solved the daily locomotive scheduling problem, we know the consists of all trains. Suppose that an outbound train l_2 at a station has the same consist as that of an inbound train l_1 at the same station and locomotives from train l_1 can feasibly connect to train l_2 . Then we can create a train-train connection between l_1 and l_2 , provided locomotives coming from train l_1 are not needed by another train departing before train l_2 . We developed an efficient algorithm (details are omitted here) to determine a maximal number of train-train connections that can be made between inbound and outbound trains. We applied this algorithm at each station to identify additional train-train connections. For the data provided to us by CSX, we were able to create about 10-15% additional train-train connections by the use of this algorithm.

8. SUMMARY OF THE ALGORITHMIC APPROACH

We now summarize our algorithm to solve the weekly locomotive scheduling problem. Figure 3 gives the steps of our algorithm.

```

algorithm locomotive scheduling;
begin
  construct the daily space-time network including light arcs (see Section 5);
  solve a sequence of LPs in the daily space-time network to determine train-train
    connections (see Section 6.3);
  solve a sequence of LPs in the daily space-time network to determine light travel
    arcs (see Section 6.4);
  solve the MIP problem (2) to determine the active and deadheaded locomotives for the
    daily locomotive scheduling problem (see Section 6.5);
  solve a sequence of (single commodity) integer programs (3) to improve the solution
    obtained by the MIP (see Section 6.6);
  use the solution of the daily locomotive scheduling problem to construct the target solution
    of the weekly locomotive scheduling problem (see Section 7);
  solve a sequence of single commodity integer programs (4) to obtain a solution of the
    weekly locomotive scheduling problem (see Section 7);
  identify additional train-train connections between inbound and outbound trains at
    each station (see Section 8);
end;

```

Figure 3. Summary of the algorithmic steps.

9. COMPUTATIONAL RESULTS

In this section, we present our computational results. We were supplied data files for several scenarios by CSX Transportation. However, for the sake of consistency, we performed most of our computational results on one scenario only. The scenario we tested comprised of 3,324 trains originating from and terminating at 119 stations. It contained 3,316 locomotives belonging to five locomotive types. 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 Megahertz. We call our software the *Advanced Locomotive Scheduling (ALS)* system.

The table shown in Figure 3 summarizes the sizes of the MIP and LP problems we solved in various stages, the number of such problems solved, and the time taken to solve these problems. The MIP problem we solved to obtain a solution of the daily locomotive assignment problem was solved heuristically, as the algorithms did not terminate even after several hours of running time. We set the CPLEX parameters so that a very good integer solution to the MIP was obtained early on by the software.

Problem	Problem Size	Solution Time
MIP model (1) for the weekly locomotive scheduling problem	The weekly network contains 8,798 nodes and 30,134 arcs, of which 3,324 are train arcs, 117 light arcs, 24,577 connection arcs, and 2,116 ground arcs. The MIP formulation contains 197,424 integer variables and 67,414 constraints.	The MIP problem is too large to be solved by the available MIP software.
MIP model (1) for the daily locomotive scheduling problem	The daily network contains 1,134 nodes and 3,965 arcs, of which 463 are train arcs, 26 light arcs, 3,178 connection arcs, and 298 ground arcs. The MIP formulation contains 22,314 integer variables and 9,974 constraints.	The MIP problem is too large to be solved by the available MIP software.
Identifying train-train connections	The daily network contains 1,134 nodes and 3,965 arcs, of which 463 are train arcs, 26 light arcs, 3,178 connection arcs, and 298 ground arcs. The MIP formulation contains 22,314 integer variables and 9,974 constraints.	We solve 132 LP relaxations to determine train-train connection arcs. The total time is 11 minutes.
Identifying light travel arcs	The daily network contains 1,134 nodes and 1,276 arcs, of which 463 are train arcs, 26 light arcs, 489 connection arcs, and 298 ground arcs. The MIP formulation contains 6,385 integer variables and 7,835 constraints.	We solve 14 LP relaxations to determine light traveling arcs. The total time is 30 seconds.
Determining locomotive flow variables	The daily network contains 1,110 nodes and 1,242 arcs, of which 463 are train arcs, 12 light arcs, 475 connection arcs, and 292 ground arcs. The MIP formulation contains 8,993 integer variables and 7,782 constraints.	We are able to get the first integer solution in 5 minutes, and a very good solution in 15 minutes.
Solving the weekly locomotive scheduling problem	For each locomotive type, there are 10,768 variables, and 14,592 constraints.	We solve 15 constrained minimum cost flow problems. The total time is 1 minute.

Figure 4. Summary of problem sizes and solution times at different steps of the algorithm.

The table shown in Figure 5 compares the solution obtained by ALS with the solution obtained by the CSX software, called *LSM (Locomotive Scheduling Model)*. The ALS solution is substantially superior to the LSM solution; it reduces the total cost substantially and dramatically decreases the number of locomotives used. The ALS solution used more than 400 fewer locomotives than the LSM solution with medium consist busting (50%) and medium light travel (0.8%). Recall from Section 1 that LSM handles consist assignment and locomotive scheduling separately, and in the locomotive scheduling phase considers each locomotive type one by one. We achieved the dramatic decrease in the number of locomotives, in comparison with the LSM, mainly by combining consist assignment and locomotive scheduling, and considering all locomotive types together in place of each locomotive type one by one.

We reduce the number of locomotives by substantially reducing the fraction of the locomotives that deadhead. Figure 6 presents the statistics on the percentage of the times locomotives are actively pulling the trains, deadheading on trains, light traveling, or idling at stations (for maintenance or just waiting to be assigned to outbound trains). We observe that in the ALS solution the percentage of the time

locomotives are actively pulling the trains is about 13.1% more than the LSM solution. Hence our solution significantly increases the locomotive productivity. We also observe that in the ALS solution, the percentage of the time locomotives deadhead on trains is considerably less than in the LSM solution.

Type of locomotives	ALS	LSM
SD40	249	498
SD50	160	171
C40	487	621
AC44	154	164
AC60	160	160
Total	1210	1614

(a)

Performance measure	ALS	LSM
Active time	44.4%	31.3%
Deadheading time	8.1%	19.6%
Idling time	46.7%	49.1%
Light traveling time	0.8%	0%
Total cost	\$9.2 million	\$13.5 million
Consist-busting	49.4%	85.0%

(b)

Figure 5. Comparison of ALS and LSM solutions.

- (a) The number of locomotives of different types used by the LSM and ALS software.**
- (b) Comparison of other statistics.**

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. Figure 6(a) shows 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. Figure 6(b) shows 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 lower locomotive utilization rate and requiring more locomotives.

Consist Busting: 50%	No Light Travel	Medium light travel	Sufficient light travel
Total number of locomotives used	1268	1210	1192
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%

(a)

	Very high consist-busting	Hight 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	1155	1165	1210	1281	1340
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%

(b)

Figure 7. Sensitivity of the solution to light travel and consist-busting.

10. SUMMARY AND CONCLUSIONS

In this paper, we present the results of a study of a locomotive scheduling problem faced by CSX Transportation, a major US railroad company. We focused on the planning version of the locomotive scheduling problem, where 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 used by the locomotive planning division at CSX. The existing software was in-house software developed over a period of ten years. While it has the substantial advantage of capturing many constraints and objectives not dealt with in the existing literature, it also has some important limitations. In particular, it was 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 upon breaking the locomotive scheduling problem into a two-step process: first solving the daily locomotive scheduling problem, and then solving the weekly locomotive scheduling problem. This approach works reasonably well for those situations where a large number of trains run most days in a 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 decision variables (i) light travel, (ii) train-train connections, and (iii) active and deadhead locomotive flow variables sequentially instead of determining them in a single model. Nevertheless, our sequential procedure heavily exploited information from a sequence of LPs, and thus relied on a system wide view of the entire planning problem. When we determine light travel arcs, we consider locomotive flow variables are also part of the LPs solved. Similarly, when we determine train-train connection variables, we again consider locomotive flow variables in the decision process. Hence 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.

From the CSX perspective, this research demonstrated that advances in OR techniques enabled a simultaneous locomotive assignment and scheduling solution on a much larger network than previously thought possible. CSX used the model in support of its operations planning effort associated with its acquisition of Conrail territory, a 30% increase in network size. Without the aid of the model, CSX would not have been confident in its ability to execute the new integrated operating plan. In addition, CSX executed a major locomotive trade with a locomotive leasing company where they swapped 156 older, low and medium horsepower locomotives for 31 newer, high horsepower locomotives. The model helped quantify the network benefits of re-centering the fleet composition. In the future, CSX is interested in expanding on this work to include locomotive fueling and servicing constraints into the algorithm. CSX also intends to introduce decision support tools into its tactical management process.

ACKNOWLEDGEMENTS:

We would like to acknowledge the help of our students, Anurag Chandra (MIT), Ozlem Ergun (MIT), Anurag Mehta (University of Florida), and Mukundhan Krishnaswamy (University of Delaware), who participated in this project and contributed in different capacities. The project was mainly funded by CSX Transportation and we gratefully appreciate the help the CSX Locomotive Division for giving us the research grant and access to their data. The project 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 to the first and third coauthors.

REFERENCES:

- Aarts, E. H. L., and J. K. Lenstra, J. K. (editors). 1997. *Local search in Combinatorial Optimization*. John Wiley & Sons, New York, NY.
- Ahuja, R. K., T.L. Magnanti, and J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, NJ.
- Ahuja, R. K., O. Ergun, J. B. Orlin, and A.P. Punnen. 2002. A survey of very large scale neighborhood search techniques. To appear in *Discrete Applied Mathematics*.
- Ahuja, R. K., J. B. Orlin, and D. Sharma. 2001a. Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem. *Mathematical Programming* **91**, 71-97.
- Ahuja, R. K., J. B. Orlin, and D. Sharma. 2001b. A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem. Submitted to *Operations Research Letters*.
- Ahuja, R. K., J. B. Orlin, and D. Sharma. 2001c. A very large-scale neighborhood search algorithm for the combined through-fleet assignment model. Submitted to *INFORMS Journal on Computing*.
- Barnhart, C., N.L. Boland, L.W. Clarke, E.L. Johnson, G.L. Nemhauser, and R. Sheno. 1998. Flight string models for aircraft fleet and routing. *Transportation Science* **32**, 208-219.
- Brannlund, U., P. O. Lindberg, A. Nou, and J. -E. Nilsson. 1998. Railway timetabling using Lagrangian relaxation. *Transportation Science* **32**, 358-369
- Chih, K.C., M.A. Hornung, M.S. Rothenberg, and A.L. Kornhauser. 1990. Implementation of a real time locomotive distribution system. In *Computer Applications in Railway Planning and Management*, T. K. S. Murthy, R. E. Rivier, G. F. List and J. Mikolaj (eds.) Computational Mechanics Publications, Southampton, U.K., 39-49.
- Cordeau, J.-F., P. Toth, and D. Vigo. 1998. A survey of optimization models for train routing and scheduling. *Transportation Science* **32**, 988-1005.
- Fischetti, M. and 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, and L. Nastansky. 1976. The engine scheduling problem in a railway network. *INFOR* **14**, 121-138.
- Forbes, M.A., J.N. Holt, and A.M. Watts. 1991. Exact solution of locomotive scheduling problems. *Journal of Operational Research Society* **42**, 825-831.
- Glover, F., and M. Laguna. 1997. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA.
- Goldberg, A.V. 1995. Scaling algorithms for the shortest path problem. *SIAM Journal on Computing* **24**, 494-504.
- Hane, C. A., C. Barnhart, E.L. Johnson, R.E. Marsten, G.L. Nemhauser, and G. Sigmondi. 1995. The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming* **70**, 211-232.
- Jovanovic, D. and P. T. Harker. 1991. A decision support system for train dispatching: An optimization-based methodology, *Transportation Research Record* **1314**, 31-40.
- Mao, C-K., and C. D. Martland. 1981. Utilization of railroad motive power: Train delay impacts and organizational considerations. *The Logistics and Transportation Review* **17**, 291-312.

- Newton, H. N., C Barnhart, and P. H. Vance. 1998. Constructing railroad blocking plans to minimize handling costs. *Transportation Science* **32**, 330-345.
- Nou, A., J. Desrosiers, and F. Soumis. 1997. Weekly locomotive scheduling at Swedish State Railways, Technical report G-97-35, GERAD, Ecole des Hautes Etudes Commerciales de Montreal, Canada.
- Smith, S. and Y. Sheffi. 1988. Locomotive scheduling under uncertain demand. *Transportation Research Record* **1251**, 45-53.
- Sherali, H. D., and A. B. Suharko. 1988 A tactical decision support system for empty railcar management. *Transportation Science* **32**, 306-329.
- Subramaniam, R., R. P. Scheff Jr., J. D. Quillinan, D. S. Wiper, and R. E. Marsten. 1994. Coldstart: Fleet assignment at Delta Air Lines. *Interfaces* **24**, 104-120.
- Ziarati, K., F. Soumis, J. Desrosiers, S. Gelinias, and A. Saintonge. 1997. Locomotive assignment with heterogeneous consists at CN North America. *European Journal of Operational Research* **97**, 281-292.
- Ziarati, K., F. Soumis, J. Desrosiers, and M. M. Solomon. 1999. A branch-first, cut-second approach for locomotive assignment, *Management Science* **45**, 1156-1168.