TRANSPORTATION SCIENCE
Vol. 42, No. 4, November 2008, pp. 492–507
ISSN 0041-1655 | EISSN 1526-5447 | 08 | 4204 | 0492

informs

DOI 10.1287/trsc.1080.0244

The Locomotive Routing Problem

Balachandran Vaidyanathan
Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611, and FedEx Express, Operations Research, Memphis, Tennessee 38125, bala.vaidyanathan@gmail.com

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

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

Given a schedule of trains, the *locomotive planning* (*or scheduling*) *problem* (LPP) is to determine the minimum cost assignment of locomotive types to trains that satisfies a number of business and operational constraints. Once this is done, the railroad has to determine the sequence of trains to which each locomotive is assigned by unit number so that it can be fueled and serviced as necessary. We refer to this problem as the *locomotive routing problem* (LRP). The LRP is a very large scale combinatorial optimization problem, and the general version that we consider has previously been unstudied and unsolved in the research literature. In this paper, we develop robust optimization methods to solve the LRP. There are two major sets of constraints that need to be satisfied by each locomotive route: (1) locomotive fueling constraints, which require that every unit visit a fueling station at least once for every F miles of travel, and (2) locomotive servicing constraints, which require that every unit visit a service station at least once for every S miles of travel. The output of the LPP is not directly implementable because the LPP does not consider these fueling and servicing constraints. The LRP considers these constraints, and its output is therefore implementable. We model the LRP by considering alternative fueling and servicing friendly train paths (or strings) between servicing stations on the network. We formulate the LRP as an integer programming problem on a suitably constructed space-time network and show that this problem is NP-Complete. This integer programming problem has millions of decision variables. We develop a fast aggregation-disaggregation based algorithm to solve the problem within a few minutes of computational time to near optimality. Finally, we present computational results and extensive case studies of our algorithms on real data provided by a major Class I U.S. railroad.

Key words: locomotive scheduling; network optimization; mixed integer programming; paths; routing; maintenance routing
History: Received: July 2007; revision received: March 2008; accepted: August 2008.

1. Introduction

Rail transport is an energy-efficient and capital-intensive means of mechanized land transport of people and cargo. Locomotives, which are power units that pull trains, represent one of the biggest capital investments of railroads. Each new locomotive unit represents a one-time investment of two to three million dollars; it also incurs periodic fueling and maintenance costs of fifty to one hundred thousand dollars a year. A locomotive (or power) plan provides an assignment of locomotives to each train in the schedule, and railroads use these plans as blueprints to manage their operations efficiently. Locomotive managers try to adhere to the operating plan as much as possible and make tactical adjustments to the plan to take into account the disruptions taking place across the network, such as train delays, train cancellations, and locomotive breakdowns. A good and feasible locomotive plan is very important for real-time locomotive assignment, and an efficient locomotive operating plan translates into an efficient real-time locomotive assignment. A feasible locomotive operating plan must satisfy a variety of operating constraints. The previous models developed by researchers to determine the locomotive operating plan ignore one important constraint, which we refer to as the fueling and servicing constraint. If this constraint is not satisfied by the model solution, the locomotive plan generated may not be implementable. In this paper, we propose algorithms to develop locomotive plans that satisfy the constraints considered earlier as well the fueling and servicing constraints.

The *locomotive planning* (*or scheduling*) *problem* (LPP) is to assign locomotive types to trains in a cyclic train schedule while honoring several operational constraints and minimizing the overall cost of assignment. In the recent literature, Ahuja et al. (2005) and Vaidyanathan et al. (2007) study the locomotive plan-

ning problem. Ahuja et al. (2005) formulated the LPP as a mixed-integer programming problem and solved it using techniques from *Very Large Scale Neighborhood Search* (VLSN), linear programming-based relaxation heuristics, and integer programming. Vaidyanathan et al. (2007) extended this approach on several dimensions by adding new constraints to the planning problem required by railroads and by developing additional formulations necessary to transition solutions of the models to practice. The models for the LPP assigned locomotive types to trains in a weekly repeating train schedule ensuring that every train got sufficient pulling power, locomotive flows were balanced, and several other operational constraints were satisfied; however, the models did not account for the fueling and servicing feasibility of individual locomotive units. In this paper, we develop methods that address the constraints of the LPP while also routing locomotive units on fueling and servicing friendly routes.

The *locomotive routing problem* (LRP) models the routing of each locomotive unit on a cyclic sequence of trains so that it can be fueled and serviced as necessary, while satisfying all other business and operational constraints considered in the LPP. We refer to the sequence of trains on which a locomotive unit travels as its *locomotive route*. Fueling feasibility requires that each locomotive route has sufficient fueling opportunities to ensure that the locomotive does not run out of fuel (*out-of-fuel* event). This translates into the constraint that every locomotive needs to have a fueling opportunity at least once for every F *miles* (e.g., 900 miles) of travel. Similarly, servicing feasibility requires that each locomotive route has sufficient servicing opportunities, and this translates into the constraint that every locomotive needs to have a servicing opportunity at least once for every S *miles* (e.g., 3,000 miles) of travel. If all the stations in the network supported fueling and servicing, any feasible solution to the LPP would be feasible for the LRP (more precisely, one could take the solution for the LPP, and use flow decomposition as in Ahuja, Magnanti, and Orlin 1993 to obtain routes for individual locomotive units). However, typically only 50% of the stations on the train network support fueling and 30% of the stations support servicing. Hence, solutions of the LPP typically cannot be easily transformed into feasible solutions for the LRP. Indeed, to the best of our knowledge, the LRP remains practically unstudied and unsolved, and our paper reports the first attempt to solve this problem.

Various locomotive scheduling models have appeared in the literature, but none of these studies directly considers the fueling and servicing requirements of locomotives. The paper by Cordeau, Soumis, and Desrosiers (1998) presents an excellent survey of existing locomotive planning models and algorithms for the LPP. There are two kinds of locomotive planning models: *single* and *multiple*. *Single locomotive planning models* assume that only one type of locomotive is available for assignment. These models can be formulated as minimum cost flow problems with side constraints. Some papers on single locomotive planning models are due to Wright (1989); Forbes, Holt, and Watts (1991); Booler (1980, 1995); and Fischetti and Toth (1997). Single locomotive planning models are better suited for some European railroads rather than North American railroads because most North American railroads assign multiple locomotive types to trains. *Multiple locomotive planning models* have been studied by Florian et al. (1976); Ramani (1981); Smith and Sheffi (1988); Chih et al. (1990); Nou, Desrosiers, and Soumis (1997); Cordeau, Toth, and Vigo (1998); and Ziarati et al. (1997, 1999). The most comprehensive and recent multiple locomotive planning models are due to Ahuja et al. (2005) and Vaidyanathan et al. (2007). The only research that considers maintenance requirements is due to Maroti and Kroon (2005). They consider the problem of routing locomotive units that require maintenance in the next one to three days and propose a multicommodity flow model to solve this problem. They do not address fueling issues. Their approach is a posteriori and tries to satisfy maintenance constraints for locomotive units close to their service thresholds. Our approach integrates locomotive plans with the fueling and servicing requirements.

The LPP (Ahuja et al. 2005; Vaidyanathan et al. 2007) is analogous to the well-studied airline fleet assignment problem (Abara 1989; Talluri 1996; Rexing et al. 2000). The LRP is analogous to the well-studied aircraft maintenance routing problem (Clarke et al. 1997; Gopalan and Talluri 1998; Barnhart et al. 1998; Talluri 1998). Integrating airline fleet assignment and maintenance routing into a single problem makes the problem very hard to solve. Hence the popular approach adopted in the airline industry is sequential; i.e., the fleet assignment problem is solved first to fix the assignment of aircraft types to flight legs, and the aircraft routing problem is solved next to route individual aircraft units. Several researchers have worked on the aircraft routing problem. Clarke et al. (1997) present a mathematical formulation for the aircraft routing problem and discuss its similarity with the asymmetric traveling salesman problem. Gopalan and Talluri (1998) and Talluri (1998) describe methods to generate aircraft routings that satisfy the four-day maintenance requirement. In another study with a different objective, Feo and Bard (1989) study the maintenance location problem to find the minimum number of maintenance stations required to meet the specified four-day maintenance requirements for a pro-

posed flight schedule. The only study that considers the integrated fleet assignment and aircraft routing problem is due to Barnhart et al. (1998). They use a branch-and-price based approach to solve the integrated airline fleet assignment and aircraft routing problem. Their integrated model is quite complex, and accordingly their solution technique is time-consuming. They report computation times of three to four hours on relatively small problems (i.e., from the railroad perspective) involving 150–200 flights.

The existing methods for solving the aircraft routing problem are not directly extendible to the LRP. There are several complexities in locomotive routing that are not present in aircraft routing. Three of the added complexities are the following: (a) fueling constraints are needed in rail transportation, but not in air transportation because all airports have fueling capabilities; (b) flight legs on an airline network are assigned to only one aircraft, whereas trains on a railroad network are usually assigned several locomotive units; and (c) aircrafts have more opportunities for repositioning to satisfy servicing constraints (such as moving an aircraft at night if it is idle), and repositioning is less disruptive in the air industry.

Even if the approach by Barnhart et al. (1998) were extendible to the LRP, we are addressing much larger problems than their approach could handle. Here we adopt a two-stage decomposition approach to solve the LRP. In the first stage, we solve the LPP, using existing methods, to obtain an assignment of locomotive types to trains (*locomotive schedule*). In the second stage, we use the methods described in this paper to solve the LRP and route locomotive units on fueling and servicing friendly cyclic routes (*locomotive routes*).

While solving the LRP, we fix the assignment of locomotive types to trains based on the output of the LPP, and we use the flexibility of varying the connections of locomotives between trains (*train connections*) to achieve fueling and servicing feasibility. We illustrate the importance of train connections through the following example: Consider a locomotive that arrives at a station on train A after traveling 500 miles since its last fueling event. Let this locomotive have the option to connect to either train B or train C. Suppose that train B has a length of 300 miles and terminates at a fueling location and that train C has a length of 500 miles and also terminates at a fueling location. If the locomotive connects from train A to C, then the total length of travel between fueling opportunities becomes 1,000 miles (>900 miles) and leads to an out-of-fuel event en route. On the other hand, if the same locomotive connects from train A to B, the total length of travel between fueling opportunities is 800 miles, and the locomotive route is fueling feasible. This example only considers a single connection between two trains; however, fueling and servicing feasibility is usually dependent on more than one connection between a set of trains. We formalize this notion by using the concept of *strings* (first described in Barnhart et al. 1998). Strings enable us to account for the fueling and servicing feasibility of a locomotive assigned to a sequence of trains. By construction, each locomotive route is made up of a sequence of strings.

After we fix the assignment of locomotive types to trains, the LRP can be solved as an independent problem for each locomotive type. The LRP can be formulated as a set partitioning problem with side constraints, with a variable for each of an exponential number of strings. A commonly used method to handle the complexity of such a problem relies on the dynamic generation of decision variables using column generation embedded in a branch-and-bound framework. This approach, also called branch-and-price, has found wide application in solving aircraft routing problems and crew scheduling problems. For branch-and-price to be successful, (1) one needs to efficiently generate columns on the fly by solving a subproblem and (2) one needs to devise a branching scheme that does not complicate the structure of the subproblem. While it is possible to efficiently generate fuel friendly strings or service friendly strings dynamically, we were unable to find efficient approaches for dynamically generating strings that are simultaneously fuel friendly and service friendly. Also, although there are efficient branching schemes for aircraft routing under the valid assumption that each flight leg is assigned one aircraft, this assumption is invalid for locomotive routing. Accordingly, we did not adopt the branch-and-price approach, instead choosing to explicitly enumerate decision variables (a process that needs to be carried out once). To handle the size and complexity of the problem, we develop an aggregation based two stage algorithm.

Our solution approach for the LRP involves the following steps. First, we enumerate alternative fueling and servicing feasible paths (or strings) between service stations in the train network using dynamic programming. Once we enumerate the strings, we formulate an integer program called *string decomposition problem* (SDP) on a suitably defined space-time network to decompose the locomotive schedule into flows on strings. The output of the SDP can then be assembled into fueling and servicing friendly cycles using the standard polynomial-time cycle decomposition algorithm (see, for example, Ahuja, Magnanti, and Orlin 1993). However, the SDP is NP-Complete, and our instances have several million decision variables. We develop an aggregation-disaggregation based method to solve this problem efficiently. Our aggregation-disaggregation method involves two steps. In the first step, we formulate and solve a much smaller aggregated SDP that can

be solved quickly. In the second step, we disaggregate the solution of the first stage. Finally, we describe necessary side constraints to the LPP that are implied by the fueling and servicing constraints. This paper makes the following major research contributions:

- We develop a modeling framework and a multistage solution approach for the LRP.
- We formulate the LRP as an integer-programming SDP on a suitably defined space-time network. We also establish that this problem is NP-Complete.
- We develop efficient dynamic programming algorithms to enumerate strings, which are the decision variables in the SDP.
- We develop an aggregation-disaggregation based algorithm to solve the SDP to near optimality within a few minutes of computational time in cases in which there are several million decision variables.
- We present extensive computational results to validate the performance of our approach. We also present case studies of our algorithms on real data provided by a Class I U.S. railroad.

We believe that this research will help railroads manage their locomotive resources more efficiently.

2. Notation and Terminology

Train Schedule: The train schedule (or train network) contains the set of trains that operate in a week. Each train has the following attributes: train ID, departure station, departure time of the week, arrival station, arrival time of the week, and duration. When carrying out locomotive planning, railroads typically consider the part of their train schedule that is periodic. We assume that the train schedule repeats periodically on a weekly basis. Henceforth we refer to trains in the weekly train schedule as *weekly trains*. Each weekly train is identified by the unique combination of its train ID and day of operation. For example, Train ID TR01, which operates on Monday, Tuesday, and Thursday, corresponds to three weekly trains, TR01-Monday, TR01-Tuesday, and TR01-Thursday. TR01 may also be referred to as a *daily train*.

Locomotive Schedule: The locomotive schedule specifies the assignment of locomotive types to all trains in the week such that all operational constraints, other than the fueling and servicing requirements, are satisfied. These constraints include pulling effort and horsepower constraints for each train, flow balance for each locomotive type, fleet-size constraints, flow upper and lower bounds on each train, and weekly repeatability of locomotive assignments.

Decomposed Locomotive Schedule: The locomotive schedule can be decomposed with respect to the different locomotive types in the fleet. The decomposed locomotive schedule that corresponds to a particular locomotive type only considers the trains that carry that locomotive type and also only considers assignments of that particular locomotive type. For example, suppose we consider a schedule with three trains: TR01, with a locomotive assignment of one unit of type A and one unit of type B; TR02, with a locomotive assignment of two units of type A; and TR03, with a locomotive assignment of two units of type B. Then the decomposed locomotive schedule corresponding to locomotive type A will contain TR01 with an assignment of one unit and TR02 with an assignment of two units, and the decomposed locomotive schedule corresponding to locomotive type B will contain TR01 with an assignment of one unit and TR03 with an assignment of two units.

Decomposed Train Schedule: The set of trains in a decomposed locomotive schedule of a particular locomotive type constitutes the decomposed train schedule of that locomotive type.

Fueling Station: A fueling station is a station where locomotives can get fueled. Each fueling station has an associated fueling time and fueling cost per gallon of fuel.

Servicing Station: A servicing station is a station where locomotives can get serviced. All servicing stations are also fueling stations. Each servicing station has a specific servicing time and servicing cost.

Train Connection: This refers to the transfer of locomotives from an inbound train at a particular station to an outbound train at the same station.

Minimum Connection Time: This refers to the minimum time required for a locomotive to connect between two trains (it varies from station to station and includes the fueling and servicing times wherever applicable).

String: A string is a connected sequence of trains $t_1 - t_2 - \cdots - t_n$; the destination of train t_i is the same as the origin of train t_{i+1}.

Fuel String: A fuel string is a connected sequence of trains such that a locomotive unit traveling on this sequence can be fueled feasibly.

Service String: A service string is a connected sequence of trains such that a locomotive traveling on this sequence can be fueled and serviced feasibly.

3. Overview of Our Approach

In Figure 1, we present our approach to solve the fueling and servicing friendly locomotive routing problem. The flowchart presents the various steps in our approach and how these steps relate to each other. In subsequent sections, we elaborate on finer details. Step 1 involves solving the LPP, and Steps 2–4 constitute solving the LRP.

As stated earlier, the locomotive schedule obtained from the LPP is one of the inputs that goes into the LRP. We solve the LPP using the algorithm of

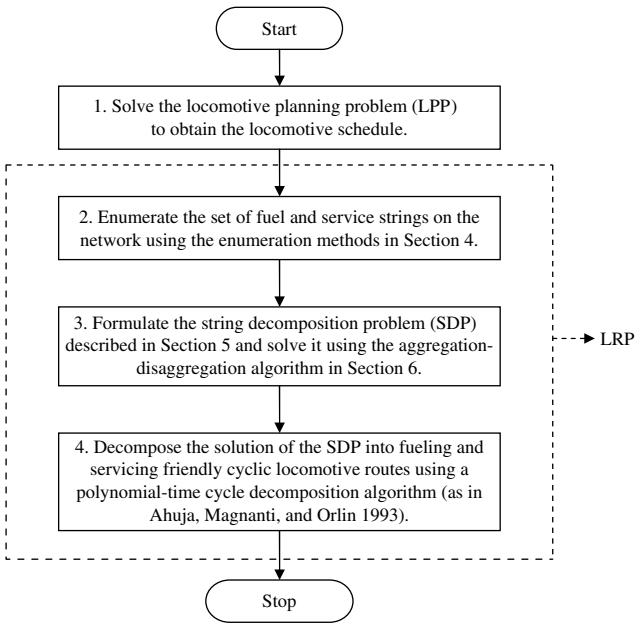

Figure 1 Flowchart of the Fuel and Service Routing Algorithm

Vaidyanathan et al. (2007), which was based on the algorithm of Ahuja et al. (2005). We then fix the assignment of locomotive types to trains. After the fixing of the assignment of locomotive types to trains, the LRP decomposes into independent problems, one for each locomotive type. Accordingly, in our description of the LRP in §§4–7, we consider problems that have only one locomotive type.

4. Fuel and Service String Enumeration Algorithms

The objective of the LRP is to route locomotive units in cycles in such a way that each unit has sufficient fueling and servicing opportunities. Consider a unit that travels on one such cycle. This locomotive will visit several stations, some that are servicing stations, some that are fueling stations, and some that are neither. Suppose this cycle is a fueling and servicing friendly cycle. Then by definition this would imply that the sequence of trains that the locomotive takes between fueling stops is a fuel string and the sequence of trains that the locomotive takes between servicing stops is a service string. This observation is the basis of our approach to solving the LRP. Our approach involves enumerating strings and then decomposing the locomotive schedule into flows on strings. In this section, we develop algorithms to enumerate strings efficiently.

In our algorithms for strings enumeration, we do not consider the compatibility of train arrival and departure times when locomotives transfer from one train to the next. Our simplification does not lead to infeasibilities because the train schedule is cyclic. For the same reason, it is possible that a train can repeat in a string. But our simplification can result in strings in which there is excessive locomotive idling or waiting time at a station. We handle this by considering the ownership cost of locomotives while solving the SDP. The objective function of the SDP ensures that large locomotive waiting will be incurred only when it is profitable.

We next define fuel strings and subsequently describe methods to enumerate them.

Fuel String

A *fuel string* is a sequence of trains $t_1 - t_2 - \cdots - t_n$ that satisfies the following properties:

- It is connected; i.e., arrival station of t_i = departure station of t_{i+1}.
- The departure station of t_1 and the arrival station of t_n are fueling stations.
- The length of the string is fewer than F miles.
- The string is minimal; i.e., the string does not pass through an intermediate fueling station.
- No train repeats in the string.

These strings are enumerated on a train schedule that is cyclic or repeats each week. Potentially, the same train could appear in a string more than once, but we explicitly prevent this from happening.

Valid Fuel Sequence and Algorithmic Logic

We define a *valid fuel sequence* as a sequence of trains that satisfies all the properties of a fuel string except that it does not terminate at a fueling station. Thus a valid fuel sequence can be obtained from any fuel string by deleting the last train(s) from it. Conversely, any fuel string can be represented as a valid fuel sequence plus one last train that terminates at a fueling station. Our dynamic programming algorithm uses these properties to enumerate fuel strings.

In the fuel string enumeration algorithm, let P denote the set of fuel strings, V^k denote the set of valid fuel sequences containing k trains, *Trains* denote the set of weekly trains, and *length*(l) denote the number of miles train l travels. We let FS denote the set of fueling stations, and F the fueling distance threshold. Also, the operation $\{v + l\}$ represents the concatenation of valid sequence v with train l to create a longer valid sequence.

algorithm *fuel string enumeration*;
begin
  $P = V^1 = \phi$;
  for each $l \in$ *Trains* such that *origin*(l) $\in FS$ and *destination*(l) $\in FS$, **do** $P := P \cup \{l\}$;
  for each $l \in$ *Trains* such that *origin*(l) $\in FS$ and *destination*(l) $\notin FS$, **do** $V^1 := V^1 \cup \{l\}$;
  for $k = 2$ to $|$*Trains*$|$ **do**
    $V^k = \phi$;
    for each valid sequence $v \in V^{k-1}$ and **for each**

```
      train l such that origin(l) = destination(v) do
    begin
      if destination(l) ∈ FS and
        if l ∉ v and if length(v) + length(l) ≤ F, then
          P := P ∪ {v + l};
      else if destination(l) ∉ FS and if l ∉ v and
        if length(v) + length(l) < F then
      V^k := V^k ∪ {v + l};
    end
    if V^k = φ then stop;
  end
end
```

The number of valid sequences and fuel strings is finite. Hence, fuel string enumeration (FSE) terminates in a finite number of steps. We now define service strings and extend the dynamic programming approach to enumerate service strings.

Service String Enumeration

We define a *service string* in a similar spirit to fuel strings. Service strings are connected sequences of trains that start and terminate at servicing stations (and do not pass through any other intermediate service station), have a length fewer than S miles, and cannot contain any train more than once. In addition, every service string also needs to be fueling feasible. Note that because all service stations in the railroad network support fueling, it follows that for the last condition to be satisfied, every service string has to be constructed from one or more connected fuel strings.

We define a *valid service sequence* as a sequence of trains that satisfies all properties of a service string except that it does not terminate at a service station. Therefore, any service string can be represented as a valid service sequence plus one fuel string that terminates at a servicing station. Our dynamic programming algorithm uses this property to enumerate service strings. The algorithm starts with a seed set of trivial valid sequences (fuel strings that originate at servicing stations and do not terminate at servicing stations) and iteratively builds longer sequences from this set. The *service string enumeration* (SSE) algorithm is similar to the FSE algorithm with the following difference: In FSE we construct fuel strings using trains as building blocks, whereas in the SSE we construct service strings using the previously enumerated fuel strings as building blocks.

5. String Decomposition Problem (SDP)

In the previous section, we described methods to enumerate service strings, which are the decision variables in the SDP. The objective of the SDP is to determine the minimum cost decomposition of a locomotive schedule into flows on service strings. Due to the manner in which service strings are constructed, it follows that a locomotive schedule, when decomposed into flows on service strings, will give fueling and servicing compliant routing of locomotive units. We formulate the SDP as an integer-programming problem on a suitably defined space-time network. This formulation is analogous to a set partitioning problem with side network flow-balance constraints. We first describe the construction of the space-time network and then the integer-programming formulation. We also show that the SDP is NP-Complete using a polynomial-time reduction from the three-partition problem.

Space-Time Network

We denote the space-time network by $G = (N, A)$ where N is the set of nodes and A is the set of arcs. For each train arrival at a service station, we create a *train-arrival* node at that station, and for each train departure from a service station, we create a *train-departure* node at that station. Once a locomotive arrives at a service station, it needs a minimum connection time before being ready for assignment on the next train. We call the time at which the locomotive is ready for assignment the *ready time*. For each train-arrival node, we assign a time attribute equal to the ready time, and for each train-departure node, we assign a time attribute equal to the departure time of the train.

We now describe the construction of arcs on the network. Let S be the set of service strings enumerated using the methods described in the previous section. Let *first*(s) denote the first train in string s and *last*(s) denote the last train in string $s \in S$. The space-time network contains a train-departure node $n_1 \in N$ that corresponds to *first*(s) and a train-arrival node $n_2 \in N$ that corresponds to *last*(s). For each string s, we construct an arc between its corresponding nodes n_1 and n_2. We denote this subset of arcs as A_s and refer to them as *string arcs*. We also construct train connection arcs (A_c) between all combinations of train arrival and departure nodes at each station to model the various possibilities of inbound locomotives being assigned to outbound trains.

The following important property directly follows from the construction of the network.

PROPERTY 1. *A locomotive that travels on a cycle in the network given above can be fueled and serviced feasibly.*

In Figure 2, we illustrate a part of the space-time network at a particular service station. This station has two incoming trains (two train-arrival nodes), two outgoing trains (two train-departure nodes), four incoming strings, seven outgoing strings, and four train connections. Note that the overall space-time

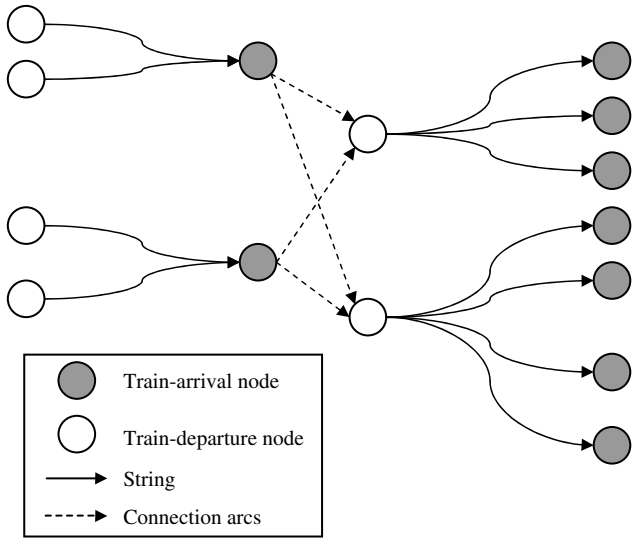

Figure 2 **A Part of the Space-Time Network at a Particular Service Station**

network contains one such component at each service station, and strings link these components to each other.

Integer Programming Formulation

We formulate the SDP as an integer-programming problem on the space-time network. Our objective is to partition the locomotive assignment on each train (locomotive schedule) into flows on the service strings that pass through it, while ensuring locomotive flow balance is conserved at each service station. We use the following notation in the formulation.

Trains: Set of weekly trains; indexed by l.

$a_{ls} = 1$ if train l is a part of string s; 0 otherwise.

f_l: Number of locomotives assigned on train l in the locomotive schedule.

O: Set of arcs that cross the count-time. The count-time is a reference time at which we count the number of locomotives used (without loss of generality, we set it to Sunday midnight).

r_s: Number of times string arc $s \in A_s$ crosses the count-time.

c_s: Cost of a locomotive unit assigned on string arc $s \in A_s$.

G: Ownership cost of a locomotive unit.

The decision variables are

x_{ij}: Denotes the flow on arc (i, j). If (i, j) is a string arc, we alternatively write the flow on the arc as x_s.

The integer program is the following:

$$\text{Min} \sum_{s \in A_s} c_s x_s + \sum_{s \in (O \cap A_s)} G r_s x_s + \sum_{j \in (O \cap A_c)} G x_j \qquad (1)$$

$$\sum_{s \in A_s} a_{ls} x_s = f_l, \quad \text{for all } l \in \textit{Trains}, \qquad (2)$$

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = 0 \quad \text{for all nodes } i \in N, \qquad (3)$$

$$x_{ij} \geq 0 \text{ integer for all arcs } (i, j) \in A. \qquad (4)$$

Constraint (2) is a set partitioning constraint that ensures the locomotive assignment on each train is distributed completely on the service strings that pass through it, and Constraint (3) is the flow balance constraint at each node. We assume that the sets of strings that can service a particular train are unrelated. A practical concern is that when a number of locomotives are assigned on a train, then uncoupling a locomotive in the center is more expensive and takes more effort than uncoupling an outer locomotive (this is referred to as consist-busting); the formulation does not address these costs. In a large sense, we avoid addressing these costs in the LRP model because they are addressed via the input to the LRP model. In our research on the LPP reported in Vaidyanathan et al. (2007), our primary objective was to minimize consist-busting. There we showed that by routing intelligently created groups of locomotives (or consists) together instead of routing individual locomotives, we can obtain a locomotive schedule with highly reduced consist-busting. In such a consist-based locomotive schedule, most trains are assigned a single consist while others have two consists. No single consist is ever busted. Even though we describe our locomotive routing algorithms using locomotive types as the commodities, we can (and do) run our algorithms on a consist-based locomotive schedule where consist-types are the commodities. In particular, we create a consist-based locomotive plan in Step 1 of Figure 1 and run our locomotive routing algorithms on this consist-based plan. Because the LPP assigns one consist to most trains in the locomotive schedule, it follows that most trains in the solution of the LRP will be serviced by only one string.

We now describe the objective function. The first term of the objective function, $\sum_{s \in A_s} c_s x_s$, represents the cost of fueling and servicing for the locomotives traveling on strings. Railroads operate in such a way that whenever a locomotive fuels, its tank is topped off. Based on this fact and the fact that the average fuel burn rate of each locomotive type is known, the cost of each service string c_s can be computed considering the cost of servicing and fueling at the final station in the string and the costs of fueling at the other fueling stations encountered enroute. The second part of the objective function, $\sum_{s \in (O \cap A_s)} G r_s x_s + \sum_{j \in (O \cap A_c)} G x_j$, minimizes the total ownership cost of locomotives by accounting for the ownership costs of locomotives traveling on overnight arcs.

The optimal solution to the SDP is a circulation. Any circulation can be decomposed into the flow

around cycles (see, for example, Ahuja, Magnanti, and Orlin 1993). Also, from Property 1 it follows that each of these cycles will be fueling and servicing friendly cycles and hence constitute a solution to the LRP.

THEOREM 2. *The string decomposition problem* (*SDP*) *is NP-Complete*.

PROOF. To show that the SDP is NP-Complete, we carry out a polynomial-time transformation from a variant of the three-partition problem, which is known to be NP-Hard (see, for example, Garey and Johnson 1979).

Input: Three sets A, B, C of n integers each, where $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $C = \{c_1, \ldots, c_n\}$, and there is an integer d.

Question: Are there n subsets $S_1, \ldots, S_n$ such that $|S_j \cap A| = |S_j \cap B| = |S_j \cap C| = 1$ and such that the sum of the values in each subset is d?

We reduce the 3-partition problem to a special case of the SDP as follows. Suppose that there are three fueling stations, labeled 1, 2, and 3. In addition, station 1 is the only service station. There are n trains from station 1 to station 2, where the jth train travels a_j miles. There are n trains from station 2 to station 3, where the jth train travels b_j miles. There are n trains from station 3 to station 1, where the jth train travels c_j miles. The maximum number of miles between servicing is d. Then the feasible strings will consist of triples of trains whose total distance traveled is at most d. Moreover, there will be a feasible solution to the SDP if and only if there is a feasible solution to the three-partition problem. □

In the next section, we address the computational complexities involved in solving the SDP and also describe algorithmic approaches to solve this problem in an efficient manner.

6. A Tractable Solution Approach: Aggregation and Disaggregation

A typical weekly train schedule has around 3,000 trains, and the railroad network has around 100 stations, out of which roughly 50% support fueling and 30% support servicing. For a problem of this size, the number of service strings runs into several millions. We did not find a direct integer-programming approach for the SDP to be viable. So we adopted a heuristic approach based on aggregation and disaggregation. In the first stage, we construct a smaller aggregated problem that can be solved quickly. In the second stage, we disaggregate the solution and create a solution for the SDP.

Aggregated Model

The aggregated problem is an approximation of the original problem. There are two major considerations in defining the aggregated problem: (1) which entities in the original problem to aggregate and (2) how to ensure that the aggregated problem is a good approximation of the original problem. We divide our discussion into two parts. First, we focus on relationship between the feasibility of the aggregated problem and the original problem and, in the process, address the first question. Next, we focus on the goodness of the aggregated model and address the second question.

We consider a set of weekly train schedule. We consider weekly trains to be *day-equivalent* if they start at the same station at the same time of day and end at the same station and if their durations are the same. The only difference is that they may operate on different days of the week. Our aggregated problem ignores the days of the week. All day-equivalent trains are aggregated together to create a set of aggregated trains. The total flow on each aggregated train represents the total flow on all weekly trains corresponding to it. The locomotive schedule obtained after aggregation is called the *aggregated locomotive schedule*, and the set of trains in this schedule constitutes the *aggregated train schedule*. We now illustrate aggregation through an example. Suppose daily train TR01 has a weekly frequency of three and operates on Monday, Tuesday, and Thursday (TR01-Monday, TR01-Tuesday, and TR01-Thursday), and suppose each of these day-equivalent weekly trains carries a flow of one locomotive of type A and one of type B. Then in the aggregated locomotive schedule, aggregated train TR01 will be assigned a flow of three units of type A and three units of type B.

Similar to the enumeration of strings from weekly trains, we can also enumerate strings using the aggregated trains; we call such strings *aggregated strings*. We can also analogously formulate an aggregated string decomposition model to decompose the aggregated locomotive schedule into flows on aggregated strings (in a similar manner to the SDP). We next summarize the steps to formulate the aggregated string decomposition model:

- Enumerate the set of aggregated service strings using the aggregated train schedule as input and using the SSE algorithm described in §4.
- Construct a space-time network as described in §5 using the aggregated strings (instead of strings) and the aggregated train schedule (instead of the train schedule).
- Formulate the aggregated SDP as described in §5 using the aggregated locomotive schedule (instead of the locomotive schedule) and the aggregated strings (instead of strings).

The aggregated model is identical in structure to the original model, but it is far smaller. The number of feasible strings decreases by several orders of magnitude. To see why, consider an aggregated string made

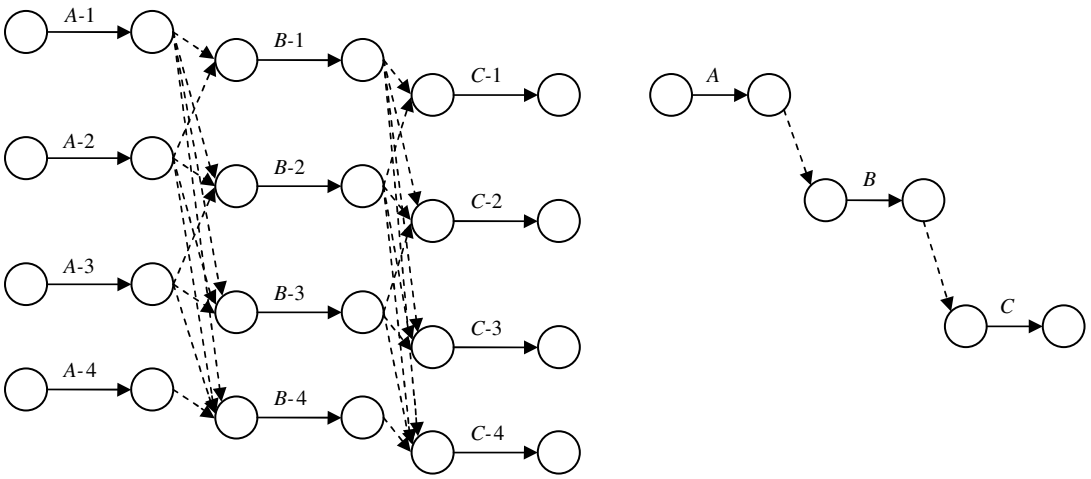

Figure 3 Illustration of Disaggregated Strings and Corresponding Aggregated String
Note. For the sake of clarity, we show only selected connection arcs.

up of aggregated trains A, B, and C. Let each of these trains have a frequency of four. Then there are $4 \times 4 \times 4 = 64$ different strings possible in the original network using the weekly trains corresponding to A, B, and C (refer to Figure 3). However, the aggregated formulation considers only one aggregate string instead of the 64 strings. Hence the flow on one aggregated string, in this case, represents the cumulative flow on 64 strings. Using this method of variable aggregation, we are able to reduce the number of variables to several thousand. In addition, there is approximately an 80% reduction in the number of constraints.

We now state an important theorem related to the aggregated problem.

THEOREM 3. *There exists a feasible solution to the string decomposition problem if and only if there exists a feasible solution to the aggregated string decomposition problem.*

PROOF. The only if part is obvious. Any solution to the string decomposition problem induces a solution to the aggregate problem. Now consider a feasible solution F to aggregated string decomposition. We can obtain a feasible solution to string decomposition by distributing the flow on each aggregated string in F on the strings corresponding to it in the original network in such a way that the flow on all weekly trains is partitioned. Because we merely redistribute the flows on the aggregated strings, the total flow entering a station and the total flow leaving a station in both the aggregated solution and the new solution will be the same. Hence, flow balance at each station is also satisfied and the result follows. □

Although the aggregated problem is equivalent to the original problem in terms of feasibility, it does not capture the costs precisely. In particular, the aggregated problem could generate bad solutions in terms of locomotive usage. For example, consider a weekly train A that arrives at a station on Monday at 10:00 A.M. and another weekly train B that departs from the same station at 11:00 A.M. on Sunday. The optimal solution of the SDP is not likely to route the same locomotive unit on these two weekly trains because the idling time of that locomotive would be extremely high (six days and one hour). On the other hand, in the aggregated problem, because we do not consider the arrival and departure days of trains (we only consider the arrival and departure times of the day), the connection time between the corresponding aggregated trains will be just one hour, and it is likely to be used in the solution. Hence we find that the aggregated problem, in such cases, may not serve as a good approximation of the original problem.

In order to improve the aggregated model, we use the *estimated connection time* for each train connection on the aggregated network. The estimated connection time between daily trains A and B represents the probable connection time if a locomotive connects between a weekly train corresponding to aggregated train A and a weekly train corresponding to aggregated train B (we cannot compute this value exactly unless both A and B have a frequency of one). We represent the estimated connection time of a train connection j by $\bar{t}_j$. Each string is made up of possibly several train connections. We compute the estimated duration of an aggregated string ($\bar{t}_s$) by adding the exact durations of the trains belonging to it and the estimated connection times ($\bar{t}_j$) for the connections that belong to it. We next describe how to use these estimated durations in the objective function of the aggregated model.

Consider the objective function of the SDP. The first term, $\sum_{s \in A_s} c_s x_s$, represents the cost of fueling and servicing for the locomotive traveling on string s. The

computation of c_s is a function of the stations encountered en route along the string, the cost of fueling or servicing at these stations, and the length of trains that belong to the string; this logic is hence directly extendible to aggregated strings. The second part of the objective function, $\sum_{s\in(O\cap A_s)} Gr_s x_s + \sum_{j\in(O\cap A_c)} Gx_j$, minimizes the total ownership cost of locomotives by counting the number of locomotives traveling on overnight arcs. In the aggregated problem, it is not possible to determine the set of overnight arcs or the number of times an aggregated string crosses the count-time (r_s). This is because the arrival and departure days of trains are abstracted in the aggregated problem. In order to counter this problem, we rewrite the objective function of the SDP in a different though equivalent form. Instead of minimizing the ownership cost of locomotives on overnight arcs, we minimize the total cost of locomotive-minutes used on all arcs. Let t_s represent the exact duration of a string arc, t_j the exact duration of a connection arc, and G' the ownership cost of a locomotive per unit time. The original objective function can be equivalently written as

$$\text{Min} \sum_{s\in A_s} c_s x_s + \sum_{s\in A_s} G' t_s x_s + \sum_{j\in A_c} G' t_j x_j. \tag{5}$$

This form of the objective function is easily extendible to the aggregated problem by using the estimated durations ($\bar{t}_s$, $\bar{t}_j$) in place of the exact durations (t_s, t_j). Our studies show that using the modified objective function in the aggregated model makes it a much better approximation of the original problem and produces a solution that can be disaggregated to obtain near optimal solutions to the SDP. We now focus on the disaggregation stage.

Disaggregation Model

The proof of Theorem 3 describes a straightforward method to obtain a feasible string decomposition starting with a solution to the aggregated problem; however, this method typically allocates too many locomotives. The objective of the disaggregation model is to disaggregate while simultaneously minimizing the number of locomotives used. (We do not consider the fueling and servicing costs in this stage because these costs remain the same.) The disaggregation problem, similar to the aggregated problem, can be solved for each locomotive type independently.

The inputs to the disaggregation problem are as follows:

- Locomotive schedule: This gives the assignment of locomotive types to weekly trains.
- Solution of the aggregated model: This gives the aggregated strings and their locomotive assignments.

The disaggregation model redistributes flow on each aggregated string on the strings corresponding to it in the original network while ensuring that the assignment on each weekly train is covered and the total number of locomotives used is minimized. We formulate the problem as an integral *multi-commodity flow problem* (MCFP) on a suitably defined network. We consider one commodity for each string with positive flow in the solution of the aggregated problem (however, the disaggregation problem is still solved as an independent problem for each locomotive type).

We denote the network as $G = (N, A)$, where N is the set of nodes and A is the set of arcs in the network. We create a *train-arrival* node corresponding to each train arrival and a *train-departure* node corresponding to each train departure. We also construct one supply node and one demand node corresponding to each aggregated string with positive flow ($x_s > 0$) and define a commodity corresponding to this string. We set the supply of each commodity at its supply node to x_s (and the supply of all other commodities to zero). Similarly, we set the demand of each commodity at its demand node to x_s (and the demand of all other commodities to zero).

We now describe the construction of arcs. For each weekly train, we construct an arc connecting its train-departure node and train-arrival node; we refer to the set of weekly train arcs as *Trains*. We also construct connection arcs (*CoArcs*) between train-arrival nodes and train-departure nodes at each station. Let *first*(s) be the set of weekly trains corresponding to the first aggregated train in aggregated string s, *last*(s) be the set of weekly trains corresponding to the last aggregated train in aggregated string s, and *trains*(s) be the set of all weekly trains corresponding to the aggregated trains in s. For each aggregated string s with positive flow ($x_s > 0$), we connect its supply node to the departure nodes of all the trains in *first*(s), and we connect the arrival nodes of all trains in *last*(s) to its demand node. The disaggregation model can be formulated as an MCFP on the network constructed above. We now introduce the notation.

K: Set of commodities indexed by k. We define one commodity for each aggregated string s with positive flow.

$k(s)$: The commodity corresponding to aggregated string s.

$s(k)$: The aggregated string corresponding to commodity k.

f_{ij}: Total locomotive assignment on weekly train $(i, j) \in$ *Trains* in the locomotive schedule.

b^k: Supply/demand vector of commodity $k \in K$.

u_{ij}^k: Upper bound on the flow of commodity $k \in K$ on arc $(i, j) \in A$.

$$u_{ij}^k = \left\{ \begin{array}{ll} 0, & \text{if } (i,j) \in \textit{Trains} \text{ and } (i,j) \notin s(k) \\ x_s, & \text{if } (i,j) \in \textit{Trains} \text{ and } (i,j) \in s(k) \\ \infty, & \text{if } (i,j) \in \textit{CoArcs} \end{array} \right\}.$$

t_{ij}: Duration of arc $(i, j) \in A$.
$\bar{N}$: Arc adjacency matrix of $G = (N, A)$.

The decision variables are:
x_{ij}^k: Flow of commodity $k \in K$ on arc $(i, j) \in A$.

The formulation is

$$\text{Min} \sum_{k \in K} \sum_{(i,j) \in A} t_{ij} x_{ij}^k. \quad (6)$$

$$\sum_{k \in K} x_{ij}^k \le f_{ij}, \quad \text{for all } (i, j) \in Trains, \quad (7)$$

$$\bar{N} x^k = b^k, \quad \text{for all } k \in K, \quad (8)$$

$$0 \le x_{ij}^k \le u_{ij}^k, \quad x_{ij}^k \text{ integer}. \quad (9)$$

The flow balance constraints (Constraints 8), and also setting flow upper bounds for each commodity on train arcs as described above, ensure that the flow on each aggregated string is redistributed completely on the strings corresponding to it in the original problem. Also, Constraint (7) ensures that the total flow on each weekly train is partitioned among the service strings that pass through it. Hence *MCFP* is a valid formulation for the disaggregation problem. The objective function minimizes the total number of locomotive minutes used.

Note that Constraints (7) are the bundling constraints, which link the commodities together and complicate the model. If these constraints were relaxed, then the problem would reduce to $|K|$ independent minimum cost flow problems. Based on this, we adopt a sequential commodity-by-commodity approach to obtain a feasible solution to the disaggregation problem. We first solve the problem for commodities that correspond to longer strings before those that correspond to shorter strings. The reason we do this is that longer aggregated strings tend to have a greater impact on the cost, and we want to optimize them first. Let $k(s)$ represent the commodity corresponding to aggregated string s. For each aggregated string s, we consider the subgraph $G^{k(s)} = (N^{k(s)}, A^{k(s)})$, which contains only those train arcs for which $u_{ij}^{k(s)} > 0$ and $f_{ij} > 0$. We solve a minimum cost flow problem on this subgraph, extract out the flows on the disaggregated strings, and update the total unaccounted flow on each train (f_{ij}). We repeat this procedure until we account for the flow on all aggregated strings.

algorithm *disaggregation*;
begin
  Let S = Set of aggregated service strings with positive flow;
  Let $SS = \phi$ represent the set of disaggregated service strings;
  Sort S in decreasing order of string length;
  for each $s \in S$ **do**
    Let $k = k(s)$; Consider the subgraph $G^k = (N^k, A^k)$;
    Solve a minimum cost flow problem for commodity k on G^k;
    Let the flow on each arc in the optimal solution be x_{ij}^k;
    Decompose x_{ij}^k into flows on paths P using a graph search algorithm; Let $SS := SS \cup P$;
    Update the uncovered flow on each train $f_{ij} := f_{ij} - x_{ij}^k$ for all $(i, j) \in Trains$;
  end for
  return SS;
end

The algorithm for disaggregation solves an independent minimum cost flow problem for each string with positive flow and is a polynomial-time algorithm. The correctness of this algorithm follows from the fact that it is a special implementation of the general method described in the proof of Theorem 3 for obtaining a feasible solution to the SDP starting from a feasible solution to the aggregated SDP.

THEOREM 4. *The disaggregation algorithm is a polynomial-time algorithm that produces a feasible solution to the string decomposition problem starting from a feasible solution to the aggregated string decomposition problem.*

We illustrate the disaggregation algorithm using the example in Figure 4. In Figure 4(a), we consider four trains T1, T2, T3, and T4. T1 has a frequency of four (T1-1, T1-2, T1-3, T1-4), T2 has a frequency of two (T2-1, T2-2), T3 has a frequency of two (T3-1, T3-2), and T4 has a frequency of one (T4-1). The output of the locomotive schedule, which is an input to the LRP, is denoted by f. Let the optimal solution of the aggregated SDP be (1) two units of flow on aggregated string S1, which contains T1, T2, T4, and (2) three units of flow on aggregated string S2, which contains T1, T3, T4. The disaggregation algorithm solves one minimum cost flow problem for each aggregated string. Because both aggregated strings have a length of three, we break ties arbitrarily and first consider string S1. In Figure 4(b), we present the subgraph and the minimum cost flow problem corresponding to S1. The optimal solution to this problem can be decomposed into two strings: (1) T1-1, T2-1, T4-1 with a flow of one and (2) T1-2, T2-2, T4-1 with a flow of one. We save these strings and corresponding flows in the set of strings SS. Next we update the uncovered flow f and set up the minimum cost flow problem corresponding to string S2 as shown in Figure 4(c). The optimal solution to this problem can be decomposed into three strings: (1) T1-1, T3-1, T4-1 with a flow of one; (2) T1-3, T3-1, T4-1 with a flow of one; and (3) T1-4, T3-2, T4-1 with a flow of one. We add

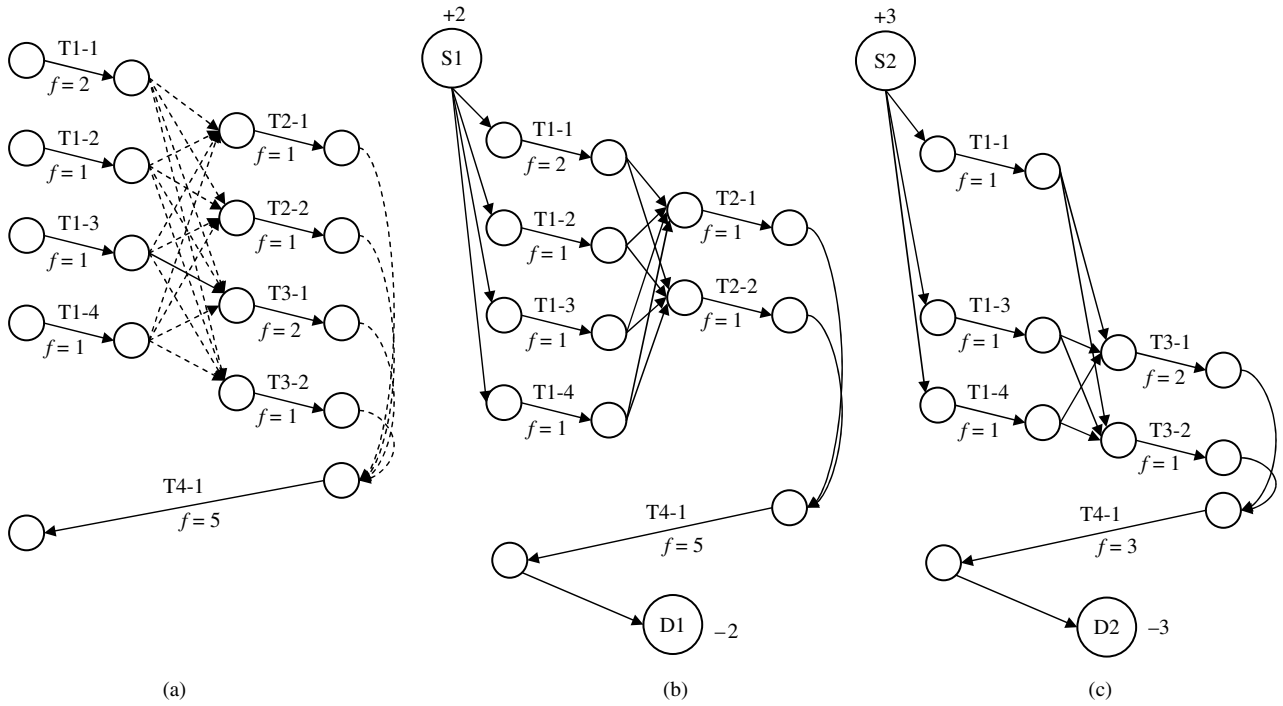

Figure 4 Example of the Disaggregation Algorithm

these strings and corresponding flow in SS, and the algorithm terminates.

In §8, we present computational results that show that the aggregation-disaggregation method solves the LRP very quickly and produces near optimal solutions.

7. Reducing Infeasibilities

In this section, we present an approach that reduces the number of fueling and servicing infeasibilities. We modify the LPP so that it incorporates some necessary service and fueling constraints. We refer the reader to Ahuja et al. (2005) and Vaidyanathan et al. (2007) for more details about the locomotive planning model; here we describe the side constraints that we add to the model.

- We only permit a locomotive connection from train A to train B if there is a feasible service string that includes trains A and B.
- If there is a unique service string s for train A with a pre determined locomotive assignment f, we require that all connections on the service string s have a flow of at least f.
- If there is a connected sequence of trains $A_1, A_2, \ldots, A_n$ such that every string containing one of them also contains the others, then we can replace the set of trains by a single train with the same origin and departure time as train A_1 and the same destination and arrival time as train A_n.

Because all of these side constraints can be applied directly to the network itself, they do not lead to an increase in running time of the planning model. Moreover, these side constraints do, in fact, lead to improved fueling and servicing friendly routing.

8. Computational Results

In this section, we provide computational results of our algorithms on several instances and case studies on a representative instance. We implemented our algorithms using Microsoft® Visual Basic programming language and tested them on the data provided by a major Class I U.S. railroad. We modeled our integer programs using the ILOG Concert Technology 2.0 modeling language and solved them using the CPLEX 9.0 solver. We conducted all our computational tests on an Intel® Pentium® 4, 512 MB RAM, 2.4 GHz processor desktop computer. All solution times reported represent the total time taken to solve the LRP, which includes string enumeration, solving the aggregated SDP, and disaggregation. We always enumerate strings on the aggregated network, and time taken for string enumeration is less than a minute for all instances.

Ideally, we should compare our solutions with those being used at U.S. railroads. However, locomotive plans that are currently being used at railroads do not consider the fueling and servicing constraints in addition to some other operational constraints. Railroads satisfy their fueling and servicing requirements by manually adjusting operations in real time. Therefore, in order to evaluate the effectiveness of our algorithm to solve the LRP, we compare our solutions with

Table 1 Performance of the Locomotive Routing Algorithm

		Locomotive planning problem		Locomotive routing problem				
Station no.	Mean tonnage	Cost ($)	No. of locos	Cost ($)	No. of locos	No. of infeasibilities	Solution time	Percentage of difference in cost
1	5,700	6,068,767	1,520	6,192,415	1,566	0	3 m 41 s	2.04
2	6,080	6,444,686	1,591	6,565,983	1,632	0	4 m 07 s	1.88
3	6,460	6,861,699	1,646	6,982,995	1,687	0	3 m 59 s	1.76
4	6,840	7,161,102	1,683	7,317,678	1,733	0	4 m 20 s	2.19
5	7,220	7,598,937	1,688	7,755,512	1,738	0	4 m 07 s	2.06
6	7,600	8,025,096	1,738	8,184,696	1,785	0	3 m 55 s	1.99
7	7,980	8,501,004	1,717	8,687,484	1,774	0	4 m 07 s	2.19

the solution of the locomotive planning models developed by Ahuja et al. (2005) and Vaidyanathan et al. (2007).

Testing the Algorithm's Performance

In this section, we test the performance of our locomotive routing algorithm on instances with different transport volumes. In order to measure the performance of our algorithms, we first solve the LPP to obtain a locomotive schedule using the methods described in Ahuja et al. (2005) and Vaidyanathan et al. (2007). Next we solve the LRP starting from this locomotive schedule and evaluate the locomotive routing cost as defined in the locomotive planning problem. We exclude the specific costs of fueling and servicing in order to make the comparison of costs between the LRP and the LPP on a similar basis. Because the LRP is more constrained than the LPP, its cost will always be greater. The smaller the difference in cost between the LPP and LRP, the better the performance of the locomotive routing algorithm. We present these results in Table 1 and Figure 5.

From our computational tests we draw the following conclusions:

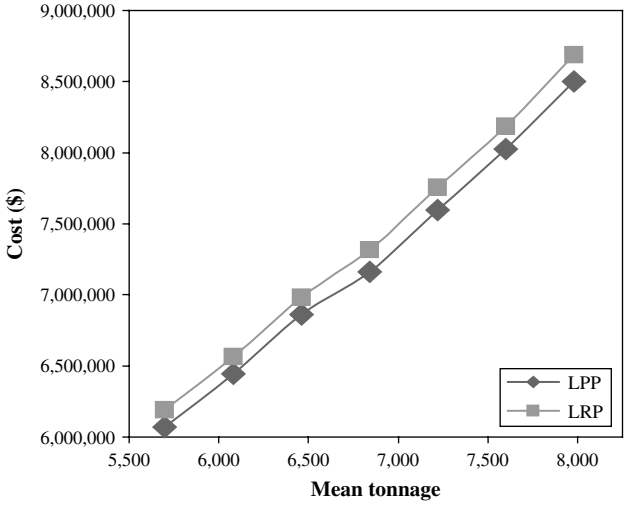

Figure 5 Performance of the Locomotive Routing Algorithm

- We are able to achieve a fueling and servicing friendly routing on all instances with less than 2.2% increase in the locomotive routing cost.
- The performance of the locomotive routing algorithm is consistent with a computational time of less than five minutes on all instances.
- The locomotive routing algorithm is able to achieve a 100% fueling and servicing friendly routing on all instances.

Case Study

In this section we present a case study to demonstrate the uses of the locomotive routing model. We perform this case study on a representative instance with 2,824 trains, 77 stations, 39 fueling stations, 28 servicing stations, and 6 locomotive types. The various aspects of the problem that we study are reported in the following sections.

Effect of Varying the Servicing Distance Threshold (S)**.** In this study, we quantify the effect of varying the maximum distance that can be traveled before servicing (S) on the locomotive routing cost. While fueling is a hard constraint because tank capacities of locomotives are well defined, servicing constraints are soft constraints because maintenance of units can, in principle, be delayed to a later time. Table 2 presents the computational results, and Figure 6 shows the relationship between the servicing distance threshold and the routing cost.

From our computational tests we draw the following conclusions:

Table 2 Effect of Varying the Servicing Distance Threshold

Station no.	Threshold S (miles)	Cost ($)	No. of locos used	No. of infeasibilities	Solution time
1	1,800	8,752,593	1,679	121	4 m 59 s
2	1,900	8,600,418	1,695	100	4 m 40 s
3	1,950	8,422,679	1,736	52	4 m 39 s
4	2,000	8,201,159	1,775	0	4 m 40 s
5	2,200	8,201,159	1,775	0	5 m 11 s
6	2,400	8,192,759	1,776	0	4 m 40 s
7	2,600	8,195,447	1,777	0	4 m 44 s
8	2,800	8,195,447	1,777	0	4 m 30 s
9	3,000	8,179,319	1,771	0	4 m 32 s

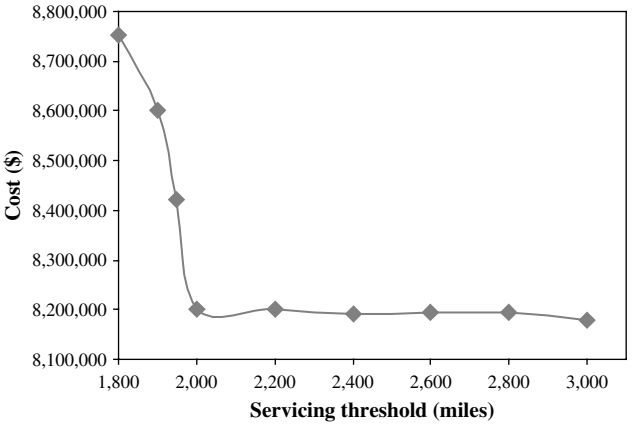

Figure 6 Solution Cost vs. Servicing Threshold

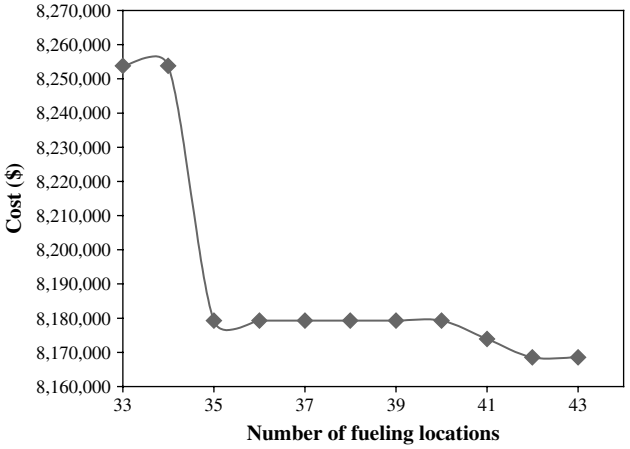

Figure 7 Solution Cost vs. the Number of Fueling Stations

• As the servicing threshold is increased from 1,800 miles to 3,000 miles, the initial increase has a large impact on feasibility and solution cost, but subsequently the law of diminishing returns is observed.

• Beyond a servicing threshold of around 2,000 miles, there is no major impact on the solution cost.

Effect of Adding/Removing Fueling Facilities. In this study, we quantify the impact of changing the number of fueling facilities on the locomotive routing cost. We start with the initial set of 39 fueling stations that currently exist on the railroad network. To this set we add (or remove) fueling stations one by one and measure the impact on the cost of locomotive routing. When we add fueling stations, we add them in decreasing order of the number of inbound trains. On the other hand, when we remove fueling stations, we remove them in increasing order of the number of inbound trains. This corresponds to adding stations that are likely to give a greater reduction of the locomotive routing cost and removing stations that are likely to not significantly increase the locomotive routing cost. Table 3 presents the computational results, and Figure 7 shows the relationship between the number of fueling stations and the solution cost.

Table 3 Effect of Varying the Number of Fueling Stations

Station no.	No. of facilities	Cost ($)	No. of locos used	No. of infeasibilities	Solution time
1	33	8,253,831	1,767	14	4 m 56 s
2	34	8,253,831	1,767	14	4 m 59 s
3	35	8,179,319	1,771	0	4 m 45 s
4	36	8,179,319	1,771	0	4 m 40 s
5	37	8,179,319	1,771	0	5 m 05 s
6	38	8,179,319	1,771	0	5 m 12 s
7	39	8,179,319	1,771	0	4 m 41 s
8	40	8,179,319	1,771	0	5 m 03 s
9	41	8,173,943	1,769	0	4 m 36 s
10	42	8,168,567	1,767	0	4 m 47 s
11	43	8,168,567	1,767	0	5 m 10 s

From our study we draw the following conclusions:

• As the number of fueling stations is increased from 33 to 43, the initial additions have a large impact on the solution cost, but subsequently the law of diminishing returns is observed.

• Solution cost remains constant for the number of fueling stations varying between 35 and 40. This suggests that we could bring down the number of fueling stations from the current number of 39 to 35 without compromising on the cost of locomotive routing. This could potentially provide significant savings for the railroad.

Effect of Adding/Removing Servicing Facilities. In this study, we quantify the impact of varying the number of servicing facilities on the locomotive routing cost. We start with the initial set of 28 servicing stations that currently exist in the railroad network. To this set we add (or remove) servicing stations one by one and measure the impact on the cost of locomotive routing. Similar to the previous case study, we add stations in decreasing order of the number of inbound trains but remove servicing stations in increasing order of the number of inbound trains.

Table 4 Effect of Varying the Number of Servicing Stations

Number of facilities	Cost ($)	No. of locos used	No. of infeasibilities	Solution time
21	8,490,764	1,748	44	4 m 22 s
22	8,319,852	1,770	16	4 m 40 s
23	8,306,412	1,765	16	4 m 26 s
24	8,182,704	1,763	4	4 m 35 s
25	8,179,139	1,771	0	4 m 47 s
26	8,179,139	1,771	0	4 m 50 s
27	8,179,319	1,771	0	4 m 36 s
28	8,179,319	1,771	0	4 m 32 s
29	8,158,151	1,759	0	3 m 27 s
30	8,147,063	1,759	0	3 m 28 s
31	8,130,935	1,753	0	3 m 29 s
32	8,112,119	1,746	0	3 m 43 s

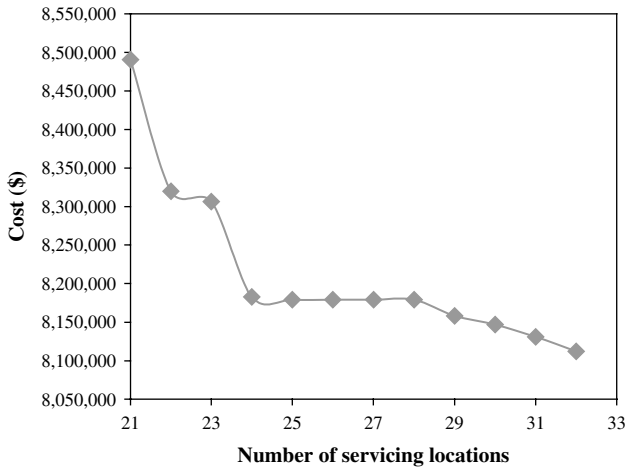

Figure 8 Solution Cost vs. Number of Servicing Stations

Table 4 presents the computational results, and Figure 8 shows the relationship between the number of servicing stations and the solution cost.

From our study we draw the following conclusions:

- As the number of servicing facilities is increased from 21 to 32, the initial additions have a large impact on the solution cost, but subsequently the law of diminishing returns is observed.
- Solution cost remains constant for the number of servicing stations varying between 24 and 28. This suggests that we could bring down the number of servicing stations from the current number of 28 to 24 without compromising on the cost of locomotive routing. This could once again potentially provide significant savings for the railroad.

9. Conclusions

Railroads use locomotive plans as blueprints to manage their resources efficiently. Our previous research on locomotive planning focused on creating locomotive plans that satisfy several business and operational constraints. Although that research has several benefits, it did not consider fueling and servicing requirements and could generate solutions that are not completely implementable. In this paper, we bridge this gap by developing optimization techniques to satisfy fueling and servicing constraints and create implementable routing of locomotive units.

Due to the inherent complexity of the integrated problem of locomotive planning and routing, we decompose our approach into two stages. The first stage involves solving the locomotive planning problem (LPP) using the algorithms described in Ahuja et al. (2005) and Vaidyanathan et al. (2007). In the next stage, we fix the assignment of locomotive types to trains and use the flexibility of train connections in order to enforce fueling and servicing feasibility. We adopt the following approach to solve the LRP: We enumerate the set of fueling and servicing feasible paths (or strings) on the network using dynamic programming algorithms and formulate the LRP as a string decomposition integer-programming problem. Due to its prohibitive size and NP-Complete nature, this problem cannot be directly solved using commercial optimization software. We handle this computational complexity by developing an aggregation-disaggregation based solution approach. Our computational results on real data demonstrate that the algorithms solve the LRP with a run-time in the order of minutes and with a less than 2.2% optimality gap. We also demonstrate the usefulness of our model as an analytical tool and give examples of the kind of insights that can be drawn from such analysis.

Our optimization methods produce practical and implementable locomotive plans. We believe that this research will help railroads manage their locomotive operations efficiently.

Acknowledgments

The authors thank Larry A. Shughart from Innovative Scheduling, who has several years of railroad experience, for helping explain the locomotive routing problem and for providing valuable business insights. The authors also thank the editors and anonymous referees for their insightful comments and suggestions, which led to a significantly improved manuscript.

References

Abara, J. 1989. Applying integer linear programming to the fleet assignment problem. *Interfaces* **19** 20–28.

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

Ahuja, R. K., J. Liu, J. B. Orlin, D. Sharma, L. A. Shughart. 2005. Solving real-life locomotive scheduling problems. *Transportation Sci.* **39** 503–517.

Barnhart, C., N. L. Boland, L. W. Clarke, E. L. Johnson, G. L. Nemhauser, R. G. Shenoi. 1998. Flight string models for aircraft fleeting and routing. *Transportation Sci.* **32** 208–220.

Booler, J. M. P. 1980. The solution of a railway locomotive scheduling problem. *J. Oper. Res. Soc.* **31** 943–948.

Booler, J. M. P. 1995. A note on the use of Lagrangean relaxation in railway scheduling. *J. Oper. Res. Soc.* **46** 123–127.

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

Clarke, L., E. Johnson, G. Nemhauser, Z. Zhu. 1997. The aircraft rotation problem. *Ann. Oper. Res.* **69** 33–46.

Cordeau, J. F., F. Soumis, J. Desrosiers. 1998. A Benders decomposition approach for the locomotive and car assignment problem. Technical Report G-98-35, GERAD, Ecole des Hautes Etudes Commerciales de Montreal, Montreal.

Cordeau, J. F., P. Toth, D. Vigo. 1998. A survey of optimization models for train routing and scheduling. *Transportation Sci.* **32** 380–404.

Feo, T. A., J. F. Bard. 1989. Flight scheduling and maintenance base planning. *Management Sci.* **35** 1415–1432.

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

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

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

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

Gopalan, R., K. T. Talluri. 1998. The aircraft maintenance routing problem. *Oper. Res.* **46** 260–272.

Maroti, G., L. Kroon. 2005. Maintenance routing for train units: The transition model. *Transportation Sci.* **39** 518–525.

Nou, A., J. Desrosiers, F. Soumis. 1997. Weekly locomotive scheduling at Swedish State Railways. Technical Report TRITA/MAT-97-OS12, Royal Institute of Technology, Stockholm, Sweden.

Ramani, K. V. 1981. An information system for allocating coach stock on Indian Railways. *Interfaces* **11** 44–51.

Rexing, B., C. Barnhart, T. Kniker, A. Jarrah, N. Krishnamurthy. 2000. Airline fleet assignment with time windows. *Transportation Sci.* **34** 1–20.

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

Talluri, K. T. 1996. Swapping applications in daily airline fleet assignment. *Transportation Sci.* **30** 237–248.

Talluri, K. T. 1998. The four-day aircraft maintenance routing problem. *Transportation Sci.* **32** 43–53.

Vaidyanathan, B., R. K. Ahuja, J. Liu, L. A. Shughart. 2007. Real-life locomotive planning: New formulations and computational results. *Transportation Res. Part B* **42** 147–168.

Wright, M. B. 1989. Applying stochastic algorithms to a locomotive scheduling problem. *J. Oper. Res. Soc.* **40** 187–192.

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

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