A Local Improvement Heuristic for the Design of Less-than-Truckload Motor Carrier Networks

WARREN B. POWELL
Princeton University, Princeton, New Jersey 08544

The load planning problem for less-than-truckload motor carriers is formulated as a fixed charge network design problem where level of service constraints are represented heuristically through a set of minimum frequencies on links. If direct service is offered between two terminals, it is required to do so at a given minimum frequency, implying not only a fixed charge from adding direct service but also a link cost function that is then flat until the flow on the link exceeds the minimum frequency. A local improvement heuristic is proposed which adds and drops links to and from the network in an intelligent sequence. After each change, the routing of the freight over the network is approximately reoptimized. Empty balancing of equipment is also handled explicitly. The approach has been applied successfully to a network with over 300 terminals; numerical tests on a 140 terminal network are reported.

Less-than-truckload (LTL) motor carriers represent the segment of the trucking industry that carries shipments that typically range from 300 to 10,000 pounds. Using trucks that carry up to 45,000 pounds, the problem these carriers face is how to consolidate the freight over the network in such a way as to minimize total transportation and handling costs while satisfying level of service constraints in each market.

An LTL network typically consists of a large number of end-of-line terminals located in various cities within a given region or, for the large national carriers, across the country. Freight is picked up in the city by a fleet of pickup and delivery trucks and carried to the end of line terminal for that city. The freight is usually then unloaded, sorted and reload onto linehaul trailers (generally a 45- or 48-foot van or, with increasing frequency, into 28-foot “pups,” two of which may be pulled by one tractor). These trailers then haul the freight over the much longer distance to a given breakbulk terminal, which is the primary sorting facility for an LTL network. A large national carrier might have 300 to over 600 end of line terminals, with approximately one breakbulk for every 20 end of lines. An excellent introduction to motor carrier operations is given by Taff.[7]

The network design problem faced by LTL carriers, usually referred to as the load planning problem, is determining how freight should be routed over the network. In practice, the problem is usually approached by motor carriers as a two-tiered problem:

1) between which pairs of terminals should direct service be offered,
2) given a set of direct services, how should the freight be routed over the network.

Offering direct service between terminals A and B implies that a trailer is loaded at A and unloaded at B with no intermediate pickups or dropoffs. Note that this does not necessarily imply that the driver actually drives direct from A to B. In union operations, the distance from A and B may exceed the maximum one driver may handle, and thus the truck may stop at one or more relay points (usually breakbulks) to change drivers.

Previous research on this problem is limited. This paper is a direct extension of the approach described by Powell and Sheffi.[4] That paper was written at a very early stage in the research, and the algorithmic approach has changed in several important ways. Barker et al.[2] used a simulation model to assist in load planning decisions, but do not present an algorithmic approach. Crainic et al.[3] describe an innovative approach for solving a similar problem for railroads. Their approach considers explicitly the frequency of service for each possible direct service as a decision variable. The procedure alternates between optimizing the frequencies while holding the flows fixed, and
optimizing the flows while holding the frequencies fixed. Roy\textsuperscript{[6]} uses a similar approach in the context of LTL motor carriers. The emphasis in this line of research is the determination of actual frequencies, and trading off costs and revenues. Balakrishnan\textsuperscript{[11]} looks at the LTL problem from a shipper's perspective and calculates a lower bound for the total costs.

The approach described in this paper uses a local improvement heuristic which, using a specific sequencing logic, works from an initial network to add or drop direct services from the network and simultaneously rerouting the flow to determine the effect of the switch on total system costs. The challenge is to develop a good sequence logic to determine the order in which to test changes, and to evaluate the system effects of a given change very quickly on a large network. Section 1 provides important background information on the problem, with special attention given to the handling of level of service. Section 2 describes the basic math programming formulation. Section 3 presents in greater detail the routing subproblem and how it is solved. Section 4 outlines how a single switch (adding or dropping a direct service) is performed, taking advantage of the structure of LTL networks. Next, Section 5 describes the logic used for sequencing the search, and finally Section 6 presents the results of a set of experiments using actual data.

1. BACKGROUND AND ASSUMPTIONS

The structure of most LTL networks and the nature of the operations provide several important simplifications which were used to advantage in the development of the algorithm. This section briefly reviews the structure of the network and some of the assumptions used.

The basic movements over an LTL network are typically made over a line operations network, illustrated in Figure 1. The length of the links in this network are governed by work rules and federal regulations. The line operations network can be viewed as an abstraction of the highway network. Separate from the line operations network is the load planning network, illustrated in Figure 2, which shows the total set of possible direct services in the network. The assumption is made that all direct services begin or end at a breakbulk; direct end of line to end of line movements are rare for large networks and can be handled on an exception basis. The cost on each load planning link reflects the actual routing of the freight over the line operations network. The network design problem consists of determining over which load planning links should direct service be offered. In the discussions which follow, a load planning link which is "in" the network represents a link over which direct service is being offered; otherwise, the link is "out" of the network.

For a given set of direct services, the next problem is to determine the routing of the freight over the network. This routing is referred to as the load plan or freight movement plan, and consists of a set of instructions of the form: freight at terminal $i$ with destination $k$ should be placed on the truck headed to terminal $j$ (regardless of the point of origin). As a result, the routing of freight from all points in the network into a destination must form a tree.

The most important modeling assumption made in this research concerns the handling of level of service. The assumption is that associated with each load planning link is a certain minimum frequency which must be maintained if direct service is offered. The effect of this minimum is to create a fixed charge for each link if it is added to the network. A minimum frequency can be viewed as a heuristic approach for ensuring an acceptable level of service in most traffic lanes, as it guarantees that freight never waits too long for the next trailer to leave. Ideally, the problem should be formulated as a detailed scheduling problem where the scheduled departure of each tractor would reflect not only a decision which balanced transportation and handling costs but also the actual level of service constraints for each shipment being carried. Two difficulties mitigate against the use of such a
detailed approach. First, given the extremely large number of origin-destination pairs, it would be computationally very slow to check the level of service for each O-D pair every time a change is made to the network. Second, it is difficult to separate “hard” service constraints, which are imposed in response to competition in certain markets, and “soft” constraints which may be changed if the nature of the flows make it expensive to maintain a given level of service.

The implication of this approach is that the load planning problem can be formulated as a cost minimization problem subject to certain operating constraints that ensure that satisfactory levels of service are provided. This fact is at the heart of the algorithm proposed in this paper.

2. THE MATHEMATICAL PROGRAMMING FORMULATION

**Define the following:**

- \( B \) = set of breakbulks
- \( E \) = set of end of lines
- \( LO \) = set of links in the line operations network
- \( LP \) = set of links in the load planning network
- \( LN \) = set of links in the load planning network over which direct service is currently being offered, \( LN \subseteq LP \)
- \( c_{ij} \) = linehaul cost per trailer (loaded or empty) from \( i \) to \( j \), \( (i, j) \in LP \cup LO \)
- \( M_{ij} \) = minimum frequency (trailers per week) from \( i \) to \( j \), \( (i, j) \in LP \)
- \( q_{rs} \) = flow (trailers per week) originating at terminal \( r \) and destined to terminal \( s \)
- \( x_{ij} \) = total flow (in trailerloads of freight per week) moving from \( i \) to \( j \), \( (i, j) \in LP \)
- \( x_{ij}^* \) = total flow (in trailerloads of freight per week) moving from \( i \) to \( j \) with destination \( s \)
- \( z_i \) = total flow handled at breakbulk \( i \), \( i \in B \)
- \( D_i \) = net supply of or demand for trailers at terminal \( i \) produced by the movement of loaded trailers over the network
- \( h_i \) = handling cost per unit of flow at breakbulk \( i \), \( i \in B \)

In addition \( \tilde{x} \) is used to denote a vector of elements, as in \( \tilde{x} = (\ldots, x_{ij}, \ldots) \).

The decision variables are as follows:

\[
y_{ij} = \begin{cases} 
1 & \text{if } (i, j) \in LN \\
0 & \text{otherwise}
\end{cases}
\]

\[
p_{ij}^* = \begin{cases} 
1 & \text{if flow at } i \text{ destined for } s \text{ must move next to } j, \\
0 & \text{otherwise}
\end{cases}
\]

\( v_{ij} \) = flow of empty trailers between \( i \) and \( j \), \( (i, j) \in LO \).

The assumption is made that a sufficient number of trucks will move on a link to carry the flow, which is to say that the actual number of trucks scheduled to move on a link is not an explicit decision variable (see Crainic et al.\[3\] for a presentation of one approach for making the frequency a decision variable). For this reason, the actual number of trucks moving on link \( (i, j) \) can be written as a function of the flow of freight on that link as follows:

\[
F_{ij}(x) = \begin{cases} 
x_{ij} & \text{if } x_{ij} > M_{ij} \\
M_{ij} & \text{otherwise}
\end{cases}
\]

Note in the definition of the frequency function \( F_{ij}(x) \) it is assumed that the number of trucks may be fractional. Recognizing that the model is a tactical planning model, the number of trucks moving can be viewed as an expectation of a random variable. In practice, when the flow is over the minimum, trucks are generally dispatched on a go-when-filled basis, and hence using \( x_{ij} \) as the actual number of trucks when \( x_{ij} > M_{ij} \) is reasonably accurate.

Finally, once the number of loaded trucks is determined for each link, we must determine how to balance the empty trucks. In short, the problem can be viewed as consisting of three stages:

1) determining \( \tilde{y} \), which links should be included in \( LN \)
2) determining \( \tilde{\rho} \), how freight should be included, which in turn determines \( \tilde{x} \), and
3) determining \( \tilde{\nu} \), the flow of empty trucks.

An exact formulation of the optimization problem is presented below, retaining the structure of the three stages. Following this presentation is a description of the procedures used to solve the overall problem, and the approximations required to keep solution times manageable.

Define:

\( \tilde{x}^*(\tilde{y}) = \text{optimal flow of freight over a given load planning network, given } \tilde{y} \)

\( \tilde{\nu}^*(x) = \text{optimal flow of empties over the line operations network for a given set of freight flows } \tilde{x} \).

The network design master problem can now be stated as follows:

\[
(NDM) \min_{\tilde{x}} \sum_{(i,j) \in LN} F_{ij}(x_{ij}^*(\tilde{y}))c_{ij} + \sum_{i \in LO} h_i z_i(\tilde{x}^*(\tilde{y})) + \sum_{(i,j) \in LO} v_{ij}^* \tilde{x}^*(\tilde{y}) |c_{ij}|
\]
subject to:

\[ y_{ij} = (0, 1) \quad \forall i, j \]  
\[ z_n(\hat{x}(\hat{y})) = \sum x_n^+(y) - \sum q_n \]  

and subject to the condition that feasible solutions exist for the subproblems which determine \( x^* \) and \( y^* \), which is guaranteed if the network \( LN \) is connected.

Problem (NDMP) proposes to design the network so as to minimize the sum of transportation, handling and empty balancing costs. Any change in the set \( LN \) defining when direct service is offered requires that the routing of the freight be reoptimized, after which the empties must also be reoptimized.

The optimal set of link flows \( x^*(y) \) represents the routing subproblem which minimizes transportation, handling and empty balancing costs. This problem can be expressed as follows:

\[
\text{(RSPE)} \min_{\varpi} \sum_{\forall i \in LN} F_{ij}[x_{ij}]c_{ij} + \sum h_n z_n(\hat{x}) + \sum_{\forall ij \in LO} v_{ij}[\hat{x}]c_{ij}
\]  

subject to Equation 4 and:

\[ \sum_i p_{ij}^i = 1 \quad \forall i, s \]  
\[ p_{ij}^i = (0, 1) \quad \forall i, j, s \]  
\[ p_{ij}^s \leq y_{ij} \quad \forall i, j, s \]  
\[ x_{ij}^s = [q_{ij} + \sum_i x_{ij}^s]p_{ij}^s \quad \forall i, j, s \quad i \in LN \]  
\[ x_{ij} = \sum x_{ij}^s \quad \forall ij \in LN. \]

Constraint (6) combined with (7) and assuming that \( c_{ij} > 0 \) for all \( (i, j) \), serve to ensure that the flow from all origins into a given destination follow a tree. As discussed earlier, this requirement reflects a routing constraint imposed by the carrier. Equation 8 implies that freight cannot be routed over a link that is not in \( LN \). Equation 9 expresses the relationship between the flow on a link with a given destination and the routing variables \( [p] \). Equation 10 gives the total flow on a link.

Finally, the empty balancing subproblem can be expressed as follows:

\[
\text{(EBSP)} \min_{\varpi} \sum_{\forall i \in LO} v_{ij}c_{ij}
\]  

subject to:

\[ \sum_k v_{ki} - \sum_j v_{ij} = D_i \]

where \( D_i \) is the net supply of trucks at node \( i \), given by:

\[ D_i = \sum_k F_{ki}(x) - \sum_j F_{ij}(x). \]

Problem (EBSP) is the only “easy” problem, in that it is a simple linear minimum cost network flow problem, which can be solved very quickly using a primal network simplex code. The next section discusses the solution of (RSPE) followed by Section 3 which outlines the local improvement heuristic for solving (NDMP).

3. THE ROUTING SUBPROBLEM

The routing subproblem is the most important component of the load planning problem due to its large size and the frequency with which it must be solved. Drawing from work presented in Ref. 5, this section presents a modified formulation of the routing and empty balancing subproblems which approximates RSPE but is much easier to solve. This new formulation is presented first followed by a description of the solution algorithm.

An Alternative Formulation of RSPE

The problem (RSPE) combines both the routing of the freight over the network and the simultaneous balancing of empty flows. As a result of the large size of the problem, however, no attempt was made to actually solve (RSPE). Instead, the routing of the freight was solved independently of the empty balancing, which was solved only after the routing problem was solved. Thus, the routing subproblem that was actually solved is given by:

\[
\text{(RSP)} \min_{\varpi} \sum_{\forall i \in LN} F_{ij}[x_{ij}]c_{ij} + \sum h_n z_n(\hat{y})
\]

subject to (4) and (6)–(10). By ignoring the empty balancing when solving (RSP) implies of course that the resulting routing of freight is not necessarily optimal. One real-world justification for solving the problem this way is that the empty balancing subproblem as presented in this research is at best an approximation of the actual routing of empties. The movement of empties tends to be somewhat unpredictable, except in certain areas with extremely large imbalance problems, and they tend to be governed by local, time-varying factors, such as the need to return a driver to his home base. In addition, the total cost of balancing empties tends to be an order of magnitude smaller than the cost of running loaded trucks. Finally, there are two extreme cases where the solution of (RSP) will also be the optimal solution of (RSPE). These cases are summarized by the following propositions:

PROPOSITION 1. Let \( \hat{x}^* \) be the solution of (RSP) and assume that \( x_{ij}^*>M_{ij}, \forall i, j \). Then \( \hat{x}^* \) is also the optimal solution of (RSPE).

Proof. Consider the problem (RSPE) and add the constraint that \( x_{ij} \geq M_{ij} \) for all \( i, j \). In this case, \( x_{ij} = F_{ij}(x) \) and it is easy to see that the terminal
deficits $D$, are given by

$$D_i = \sum_x x' - \sum_j x_j$$  \hspace{1cm} (15a)$$

$$\delta = \sum q' - \sum q_j.$$  \hspace{1cm} (15b)$$

Now the deficits $\{D_i\}$ are independent of $x$ and therefore (RSEP) decomposes into the two independent subproblems (RSP) and (EBSP). Finally, under the assumption that $x^* > M$, the added constraint that $x_j > M_j$ is unnecessary, which completes the proof. \(\square\)

Proposition 1 claims that when flows are very heavy, (RSEP) decomposes into (RSP) and (EBSP). The other extreme is when flows are very light, as stated in Proposition 2.

**PROPOSITION 2.** Let $\hat{x}^*$ be the solution of (RSP) and assume that $x^*_j < M_j, \forall i, j$. Then $\hat{x}^*$ is also the optimal solution of (RSEP).

**Proof.** Add the constraint the $x_j \leq M_j$ to RSPE. In this case, $F(x) = M$ and the terminal deficits $D$, are now purely a function of $M_j$. The remainder of the proof parallels that for Proposition 1. \(\square\)

Having argued that solving (RSP) and (EBSP) separately will provide at least a good approximation for (RSEP), we are still faced with the problem of solving (RSP). The difficulty is that the set of routing decision variables $\{p\}$ can be extremely large. A network with $N$ terminals will have approximately $N^3$ of these variables, of which approximately $N^2$ will take on the value 1 while the rest will be zero. Since $N$ may be as large as 500 for the largest motor carriers, an algorithm which deals explicitly with each variable would be extremely slow. As an alternative, an equivalent formulation is used which takes advantage of the structure of the problem, which suggests that routing flow over the least unit cost path will be accurate for most of the flow. In this context, “cost” is defined as the total linehaul and handling cost per trailer. There will, however, be specific instances where such simple logic is not sufficiently accurate, and hence the notion of a shortest path subject to routing overrides is introduced. A routing override is defined as follows:

$$o^*_j = \begin{cases} 
1 & \text{if freight at } i \text{ destined for } s \text{ must move next over the link to } j, \\
0 & \text{otherwise.} 
\end{cases}$$  \hspace{1cm} (16)$$

The set of overrides $\{o\}$ is as large as the set routing variables $\{p\}$, but the argument is that the large majority of overrides will be zero, allowing the freight to follow the “natural” path given by the least linehaul and handling cost per trailer.

The routing subproblem with overrides can now be formulated as follows:

(RSPO) $\min_{o, q} \sum_{o \in L} F(o) x_{o^*} c_o + \sum_n h_n x_n (\hat{x}^*)$  \hspace{1cm} (17)$$

subject to:

$$\sum_j o^*_j \leq 1 \ \forall i, j, s$$  \hspace{1cm} (18)$$

$$o^*_j = (0, 1)$$  \hspace{1cm} (19)$$

and where $\hat{x}^*$ is the solution of:

(LCPO) $\min_{o} \sum_{o \in L} x_o c_o + \sum_n h_n z_n$  \hspace{1cm} (20)$$

subject to:

$$\sum_i x_i - \sum_j x^*_j = b_j \ \forall j, s$$  \hspace{1cm} (21)$$

$$x_i \geq [q_j + \sum_k x^*_k] o^*_j \ \forall i, js \ \forall j \in L$$  \hspace{1cm} (22a)$$

$$Z_n = \sum x_m - \sum q_m$$  \hspace{1cm} (22b)$$

where

$$b_j = \begin{cases} 
\sum q_m & \text{if } j = s \\
-q_m & \text{otherwise.} 
\end{cases}$$  \hspace{1cm} (23)$$

LCPO routes flow over the least cost path subject to certain overrides. Equation 21 with 23 represents standard flow conservation constraints, while (22a) enforces the overrides. If $o^*_j = 1$, then flow at node $i$ headed for destination $s$ must move from $i$ to $j$, where $(i, j) \in L$. If $o^*_j = 0$, then the flow from $i$ to $s$ may follow the least cost path, defined in terms of the linehaul costs per trailer $[c]$ and the handling costs at the breakbulks $[h]$.

Several observations are in order. First, if all the overrides are zero, then (LCPO) is a simple least cost shortest path problem which decomposes for flow into each destination. Second, if all the overrides are zero and $x^*_j > M_j, \forall i, j$, then the optimum set of overrides, $\hat{o}^*$, are all zero, and hence (LCPO) solves (RSPO). Third, the set of flows $\hat{z}^*$ will automatically satisfy the tree constraints implied by (6) and (7). Fourth, if every override is specified, then $\hat{z}^*$ is completely determined by the set $\{o\}$ and (LCPO) becomes trivial. At this extreme, (RSPO) is exactly equivalent to (RSP).

Problem (LCPO) is solved by first observing that it decomposes into a separate subproblem for flow into each destination, and that the flows into each destination will follow a least cost path subject to overrides. Each subproblem can be efficiently solved using a specialized label correcting algorithm.

Aside from simply reducing the number of integer variables that must be handled explicitly for solving (RSP), the usefulness of the overrides becomes even more apparent in the context of the actual network design problem. Assume that for a given set of links
in the load planning network, \( LN \), (RSPO) has been solved within a sufficient degree of optimality. Next assume that a link is added to \( LN \) as part of the network design process. Under the original formulation of (RSP), the link that was just added would not attract any flow until the set of routing variables \( \{p\} \) was explicitly manipulated. Using (RSPO), on the other hand, it is possible to allow any flow not specifically constrained by overrides to use the new link if this new link offers a lower cost path. As is shown below, such rerouting of flows based purely on the cost per trailer can be performed extremely quickly. Furthermore, since experiments indicate that 10–20% of the flow is directly affected by the overrides, it is often possible to reroute flow, in effect, resolving (LCPO), without resolving (RSPO). The overrides, thus, become a mechanism for fine tuning the routing of the flows, while solving (LCPO) without changing the overrides can often be used to obtain a good approximation of the effect a change in the network will have on the routing of the freight.

**Solution Procedure**

The only reason that overrides are required is that solving LCPO without any overrides could (and generally will) produce some links where \( x_{ij}^{\ast \ast} < M_{ij} \). If LCPO is solved without overrides, the result may be links where \( x_{ij} < M_{ij} \) which should be carrying more flow; correspondingly, there may be other links were \( x_{ij} > M_{ij} \) which should be carrying less flow. The error, of course, is that LCPO does not accurately model the marginal cost of each unit of flow. The procedure for handling the overrides, then, works on one link at a time where the flow is below the minimum, and attempts to manipulate the overrides in such a way as to increase the flow on the link while decreasing total system routing costs.

First define:

\[
m_{ij} = \begin{cases} 
c_{ij} & \text{if } x_{ij} \geq M_{ij} \\
0 & \text{if } x_{ij} < M_{ij} \end{cases}
\]  

\( P(s) \) = set of links \( (i, j) \), where \( (i, j) \in LN \), which are in the tree describing the routes into destination \( s \)

\[
\pi_{i}^{\ast} = \text{average cost from node } i \text{ to destination } s \\
\quad \text{along the current path} \\
\quad = \pi_{i}^{\ast} + c_{ij} \quad (i, j) \in P(s)
\]  

\[
\hat{\pi}_{ij}^{\ast} = \text{average reduced cost from } i \text{ to } j \text{ for destination } s \\
\quad \text{along current path} \\
\quad = \pi_{i}^{\ast} + c_{ij} - \pi_{i}^{\ast} \quad (i, j) \in LN
\]

\[
\eta_{i}^{\ast} = \text{marginal cost from node } i \text{ to destination } s \text{ along the current path} \\
\quad = \eta_{i}^{\ast} + m_{ij} \quad (i, j) \in P(s)
\]

\[
\hat{m}_{ij}^{\ast} = \text{marginal reduced cost from } i \text{ to } j \text{ for destination } s \\
\quad = \eta_{i}^{\ast} + m_{ij} - \eta_{i}^{\ast} \quad (i, j) \in LN
\]

\( t_{i}^{\ast}(\delta) \) = next transfer node from \( i \) to destination \( s \) along current path, determined from solving LCPO, given a current vector of overrides \( \delta \).

\( u_{i}^{\ast} \) = total flow moving through node \( i \) with destination \( s \).

A family of procedures are used which are denoted by IFOL-\( n \), for increase flow on link (from \( i \) to \( j \)). The parameter \( n \) controls the search for flow to divert over link \( (i, j) \). IFOL-0 is used to take flow currently passing through node \( i \) to different destinations \( s \), but where \( t_{i}^{\ast} \neq j \), and divert it along the link \( (i, j) \). The second procedure, IFOL-1, looks at flow that is passing through node \( n \), where the link \( (n, i) \in LN \), to destinations \( s \) where \( t_{n}^{\ast} = j \) but \( t_{n}^{\ast} \neq i \). Additional procedures could be introduced which considered flow moving through nodes that are several links away from node \( i \), but the two described below have proved satisfactory. The logic is best illustrated in Figure 3 which shows a tree into destination \( s \). The links \( (i, j) \) and \( (n, i) \) are in the network but are not being used. Procedure IFOL-0 would take the flow at \( i \) and try to

---

Fig. 3. Traffic diversion over link \( (i, j) \) into destination \( s \).

---
force it over the link \((i, j)\). This would first require that the marginal cost over the new path be shorter, and that total costs be reduced if the switch actually is made. Assume this occurs. IFOL-1 would then try to take the flow at \(n\) and route it over the link \((n, i)\), from where it would then move over the link \((i, j)\).

First define:

\[
D = \{ s \mid t_s^* \neq j \}
\]

\[
T_i(s) = \text{best downstream node from } i \text{ to destination } s \text{ ignoring overrides}
\]

\[
= \{j \mid \pi_j^* + c_j = \min_k [\pi_k^* + c_k], \quad (i, j) \in LN\}.
\]

The steps of procedure IFOL-0 \([i, j, D]\) are described as follows. For a given link \((i, j)\) and destination set \(D\), choose a particular destination \(s \in D\) and calculate \(u_{ij}, \eta_{ij}, \eta_{ij}^*\) and \(m_{ij}^*\). If \(m_{ij}^* < 0\), then routing an additional unit of flow routed along \((i, j)\) to destination \(s\) should reduce costs. The estimated savings of such a switch is \(u_{ij}^* \cdot m_{ij}^*\). Store this savings and the associated destination and move to the next destination. After scanning all destinations, sort the savings and in order of the savings add or drop overrides as necessary to obtain the necessary change in routing. After each change, determine the actual change in total costs and keep the change only if costs were reduced.

The next procedure, IFOL-1 \((i, j, D)\), attempts to divert flow moving through nodes adjacent to node \(i\), to destinations \(s\) where the current path from \(i\) to \(s\) already moves over link \((i, j)\). The steps in this procedure can be described as follows. For a given destination \(s\), with \(t_s^* = j\), look at all nodes \(r\) such that direct service is offered from \(r\) to \(i\) but \(t_r^* \neq i\). As with IFOL-0, compute \(u_{ir}, \eta_{ir}, \eta_{ir}^*\) and \(m_{ir}^*\). If \(m_{ir}^* < 0\), store the estimated savings of \(m_{ir}^* \cdot u_{ir}^*\) and the associated destination. Repeat for all destinations \(s\) as was done in IFOL-0, sort the list and then implement them, testing for actual cost savings as the procedure progresses. Repeat this entire process for all valid origins \(r\).

It is not hard to see that additional procedures, IFOL-N, could be devised for diverting flow over \(N\) links before arriving at node \(i\). The structure of most LTL networks, however, is such that this is generally not necessary nor, for certain practical reasons, desirable.

The procedures IFOL-0 and IFOL-1 are used in tandem to increase flow on a link that is currently below the minimum. The general procedure, IFOL \((i, j)\), proceeds first by invoking IFOL-0 \((i, j, D)\), where \(D = \{ s \mid t_s^* \neq j \}\). Next the procedure invokes IFOL-1 \((i, j, D)\), with \(D = \{ s \mid t_s^* = j \}\). Thus IFOL \((i, j)\) is a method for refining the optimization of RSPO.

Procedure IFOL is used in two ways. First, and most important, it will be used directly in the local improvement heuristic described in the following section. If a link is added to the network and the flow attracted to the link is below the minimum, IFOL will be used to determine if any additional flow can be attracted which may further reduce system costs. Second, it is periodically necessary to sweep over all the links in the load planning network which have flows below the minimum to determine if better routings may be found for a given set of direct services. Such a search is conducted by sorting on the difference \(M_{ii} - x_{ii}\) for links where \(x_{ii} < M_{ii}\). Starting with the link with the largest value for \(M_{ii} - x_{ii}\), IFOL is called for each link to determine if further improvements may be found. This approach represents a local improvement heuristic for solving (RSPO).

4. THE LOCAL IMPROVEMENT HEURISTIC FOR NDMP

The procedure for optimizing NDMP begins with an initial feasible solution and then proceeds to sequentially add and drop links to and from the load planning network. Each local change, in the form of adding or dropping a link, requires that RSP be reoptimized at least approximately. After the flow is rerouted, the change in total system costs is determined. If costs are reduced, the change is made; otherwise the network is left intact. As it will be necessary to test hundreds of possible changes before a local minimum is found, it is important that each change to the network be fully evaluated extremely quickly. This section describes the basic mechanics for recosting a change in the network.

Given that a link has been added or dropped from the network, the problem is to reoptimize RSP and EBS. The empty balancing subproblem is the easiest of the subproblems and can be solved from scratch within a fraction of a second even for very large networks. For this reason, no special procedures were used to further streamline this problem beyond the use of an efficient network simplex code. The considerably more difficult problem is the reoptimization of RSP following a change in the network. As described in Section 3, RSP is solved by solving RSPO with LCPO as a subproblem. Completely reoptimizing RSPO after each link is added or dropped from the network, however, would be computationally infeasible for large networks (over 300 terminals). Different procedures were used for adding and dropping links which take advantage of the structure of the network.

DroppingLinks

After dropping a link LCPO is reoptimized without changing any overrides (with the exception, of course,
of eliminating overrides which force freight over the link being dropped. Since resolving LCPO alone is itself a large task, special procedures were developed which take advantage of the structure of the load planning network. Specifically, most of the links are into and out of end of lines, and freight cannot be transferred at end of lines. Assume that the trees from each node into each destination, |τ|, are stored at all times as are the costs along the trees, |π|. Recognizing that |τ| is the current basis for LCPO, the problem becomes one of simply updating the basis. If link (i, j) is being dropped, the manner in which the basis is updated depends on whether i, j, or both i and j are breakbulks.

Breakbulk to End of Line. Dropping a link into an end of line requires only that the shortest path into the end of line (node j) be computed. Freight is subtracted from the current path as defined by |τ| and is added to the new path, using the newly calculated tree.

End of Line to Breakbulk. As a result of the overrides, the paths from an origin to all destinations do not follow a tree. For this case, examine the tree into each destination s which includes the link being dropped. The problem then becomes one of finding the best breakbulk k where (i, k) ∈ LN and where π_k + c_{ik} is a minimum.

Breakbulk to Breakbulk. Consider each tree which includes (i, j) and recalculate the entire tree. Determine which origin-destination pairs are actually affected, and reroute the flow by subtracting the flow from the old path and then adding it to the new path.

Following the rerouting of the freight, EBSP is resolved and the entire system is recosted.

Adding Links

Adding links is considerably more complex than dropping them since first it is necessary to determine which trees would use a link if it were added to the network. Then it is necessary to adjust the overrides, since the total flow attracted to a new link may be restricted due to overrides. Alternatively, if the flow attracted to the new link is less than the minimum, additional flow may be attracted by adding more overrides. Thus the procedure for adding links consists of two stages: reoptimizing LCPO using the current overrides, and then a partial reoptimization of RSPO.

Stage I. Reoptimizing LCPO

After adding (i, j) to LN, the first step is to determine which flow will be attracted to the new link without changing the current set of active overrides. This step parallels that for dropping links. In the discussion that follows, let \( \hat{c}_{ij} \) be the average reduced cost over link (i, j) for destination s using the current set |π|, as defined in Section 3.

Breakbulk to End of Line. Recompute the tree into j and reroute the flows as described above.

End of Line to Breakbulk. For each destination s where \( \hat{c}_{is} < 0 \) and \( \pi_{is} = 0 \) for all \( k \), switch the routing from i to s over the new link and reroute the flows.

Breakbulk to Breakbulk. For each destination s where \( \hat{c}_{ij} < 0 \) and \( \pi_{is} = 0 \) for all \( k \), recompute the tree into destination s and reroute the flows that have been affected.

Note that prior to adding a link from i to j it is normal to set \( \pi_{is} = 0 \) for all \( k \) to ensure that freight at i destined to j is allowed to move over the link (i, j).

Stage II. Reoptimizing RSPO

After LCPO has been solved, the next step is to determine if any overrides could be changed (either added or dropped) to refine the solution. Since it will not be possible to reoptimize RSPO completely, procedure IFOL (i, j) is used for increasing the flow only on link (i, j). This step is used only if the flow on link (i, j) after Stage I is less than the minimum \( M_{ij} \). After Stage II has been completed, the system is recosted to determine if a net reduction has been achieved. If total costs increase, then the link is dropped back out of the network using the same process described above for dropping links. Note, however, that changes in the set of active overrides that were made in Stage II are not restored to their original values. Overrides which were inserted to force flow the new link would be ignored, and freight would be allowed to follow the shortest path. This approach was adopted to simplify the programming of the algorithm, but implies that if a link is added and then dropped back out, the network is not necessarily restored to its original state. This did not prove to be an important problem, but did occasionally produce situations in which total costs would increase following the attempted addition of certain links, particularly when the network was close to a local optimum. Periodically, procedures were used to eliminate overrides which were not being used (i.e., overrides which forced freight over links not in the network), after which all overrides were reoptimized by applying procedure IFOL to all links in the network below the minimum.

This section has described how a single local improvement is performed. The next step is to determine in which order the algorithm should attempt to add and drop links.

5. SEQUENCING THE SEARCH

The solution of NDMP is achieved by considering local improvements to the network, where a local improvement is in the form of adding or dropping a link. As described in Section 4, each addition or
deletion of a link requires at least a partial reoptimization of the routing subproblem RSP as well as resolving the empty balancing subproblem EBSP. The new total costs are then computed and compared to the current costs, and if a reduction is found, then the change is “kept.” The nature of the algorithm is such that a local minimum will be found, since at every step the objective function either stays the same or goes down. Clearly, the order in which improvements are tested can affect the quality of the final solution. For this reason, it is important to have a well defined logic which determines in which order potential changes are evaluated.

The sequencing logic is defined in two levels. Level I defines sets of links which are visited in a given order, while level II determines the order in which links in a given set are examined. The different sets of links are defined as follows:

1) End of line to breakbulk
2) Breakbulk to end of line
3) Breakbulk to breakbulk.

Thus, one pass of the algorithm might try to drop breakbulk to breakbulk links, while another pass might try to add breakbulk to end of line links. Of course, it is only possible to drop links which are currently in the load planning network (as defined by the set LN), and vice versa. Since it is possible to add or drop links in any of the three sets, there are in effect six sets of changes to be considered. These sets are denoted DEB, DBE, DBB, AEB, ABE, ABB, where the first letter of the code (A or D) refers to adding or dropping links, while the second two letters describe the type of links (EB is end of line to breakbulk, BE is the reverse, and BB is breakbulk to breakbulk). The level I sequencing problem, then, is to determine in which order these sets should be visited. Different sequences are described and tested in Section 6.

The level II sequencing problem is one of determining how to sequence the search within a given set. For the purpose of dropping links, the experiments in Powell and Sheffi[4] demonstrated that sorting links in order of the flow they are carrying, starting with the link with the least flow, provides a very good mechanisms for identifying improvements. This logic is used again in this research.

The more difficult problem is determining in which order to add links. Since typically less than 10% of the links in the set LP will be included in LN, there are many more potential additions to be made than deletions. For this problem, a slight modification of the logic used in Ref. [4] is used. Defining \( \hat{c}_v^* \) and \( u_v^* \) as before, define also:

\[
\mu_v^* = \begin{cases} 
  u_v^* & \text{if } \hat{c}_v^* < 0 \\
  0 & \text{otherwise.}
\end{cases}
\]

(29)

\( \mu_v^* \) is an approximation of the amount of flow that will be attracted to link \((i, j)\) destined for \(s\) if \((i, j)\) is added to the network. The total estimated flow is given by

\[
\mu_v = \sum \mu_v^*.
\]

(30)

Note that the estimate of the new flow on link \((i, j)\) does not take into account overrides that might restrict flow from using the new link. This logic approximates not only the reoptimization of LCPO but also anticipates potential changes in overrides that may be made when procedure IFOL is applied to the new link.

The approximate savings that will be realized if \((i, j)\) is added is now given by:

\[
w_v = \sum [\mu_v^* \cdot (\eta_v^* - \eta_v)] + c_v F_v(\mu_v).
\]

(31)

The first term on the right-hand side of (31) is an approximation of the changes in costs that will occur on links currently in the network as a result of the flow being rerouted over the new link. The second term is an estimate of the total costs for the new link, taking into account the fixed change that will occur if the link is added. The sequencing of potential link additions may now be made by sorting links in terms of \(w_v\) and beginning with the link with the largest estimated savings. The next section demonstrates the effectiveness of this logic in the context of a numerical example.

6. NUMERICAL EXPERIMENTS

On reviewing the algorithms described in the previous sections and, in particular, the general strategy of the local improvements heuristic, several questions arise in the context of an actual application of the approach to a realistic network. What is the overall speed of the procedure in terms of CPU time when applied to an actual network? What is the rate at which the procedure reaches a local minimum? What is the importance of the level I sequencing logic? How effective are the rules used for the level II sequencing logic? What is the value of the override manipulation routines as compared to the simple logic of optimizing RSPO without any overrides?

A series of experiments were conducted using a network extracted from the actual network for Ryder/PIE Nationwide, a national motor carrier with over 300 terminals. The network that was extracted included 12 breakbulks with 128 end of lines covering primarily the eastern United States. The purpose of using the smaller network is 3-fold: to maintain confidentiality of the data, to reduce the cost of running the experiments and to obtain a network that is more typical of most motor carriers. A starting solution was obtained by using the actual freight routings used by Ryder/PIE over the smaller network. Note that by paring off over half the terminals, the freight flows
are greatly reduced and hence using actual freight routings with reduced freight levels implies that the starting solution cannot be used as an indication of how well the carrier is currently routing the freight. Thus “savings” in costs over the base case are not meaningful, but comparisons can be made in the quality of the solutions obtained using different methods starting from the same initial solution.

The first experiments considered two alternative level I sequences. The first consisted of the sequence \( \text{DEB, DBE, DBB, ABB, ABE and AEB} \). The second sequence is simply the reverse of this sequence. One complete pass through all six sets is referred to as one pass, and three passes were used for each sequence. The logic behind the first sequence is to try to maximize the flows into the breakbulks to create as much consolidation of flows as possible. Thus \( \text{DBE and DEB} \) attempt to drop links that go directly to or from end of lines to breakbulks. Only after as many of these have been dropped as possible do we then execute \( \text{DBB} \) to drop unnecessary breakbulk to breakbulk links. Then we try to add breakbulk to breakbulk links (\( \text{ABB} \)), followed by \( \text{ABE and AEB} \). The set \( \text{AEB} \) is performed last since the handling costs at the origin increase with the number of outbound links, reflecting increased sorting costs. As a result, it seems reasonable that the number of links outbound from end of lines should be held to a minimum to encourage adding directs inbound to end of lines. Thus by considering the set \( \text{AEB} \) last, only those outbound links which really make sense will be added to the system.

The experiment was run as follows: First, the “base case” costs and other operating statistics were calculated, including the travel times between each origin and destination. These travel times were used as the base level of service. Then, all overrides were dropped to create an unencumbered initial solution. The optimization was then run as a sequence of optimizing overrides, followed by a complete pass through the network adding and dropping links as specified in the level I sequence. Although some manipulation of overrides is performed when some links are added, a complete “reoptimization” of overrides is performed after each pass through the network.

The results of this first experiment are summarized in Tables I and II. Note that the first sequence does slightly better, although the difference is very small, suggesting that the quality of the solution is not particularly sensitive to the level I sequence. Also shown in each table is the CPU time required for each pass, as well as several operating statistics. These include the percent of freight for which the travel times are one day or more over the travel times for the base case, the “transfer rate,” which is the average number of times each shipment is handled at a breakbulk multiplied times 100, and finally the system load average in pounds per 28-foot trailer (with an assumed capacity of 20,000 pounds). These statistics indicate that the final network uses less consolidation of freight through the network (as indicated by the lower transfer rate) with a higher average load average. In both cases, however, level of service is noticeably worse.

The next question that was addressed is the value of the routing overrides, as opposed to solving the routing subproblem using LCPO alone without overrides. Table III shows the results of an experiment with the same sequence used in Table I but without introducing or optimizing overrides. While clearly a procedure with the ability to optimize overrides should outperform a procedure which is not given that opportunity, the size of the difference is not clear, since this limitation can be partially overcome by running fewer direct schedules and therefore having more schedules running with more freight than the minimum. The results of the experiment suggest that while this certainly occurs, overall the routing overrides do provide a noticeable additional reduction in costs. The possibility of adding overrides only after the load planning network is optimized might reduce this gap.

### Table I

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Savings</th>
<th>Percent Over Base</th>
<th>Transfer Rate</th>
<th>Load Average</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>825,562</td>
<td>4.6</td>
<td>120.7</td>
<td>15,826</td>
<td>92.6</td>
<td></td>
</tr>
<tr>
<td>Oversides</td>
<td>815,566</td>
<td>-10.996</td>
<td>9.6</td>
<td>120.7</td>
<td>16,238</td>
<td>92.6</td>
</tr>
<tr>
<td>Pass 1st</td>
<td>781,190</td>
<td>-34.466</td>
<td>23.1</td>
<td>107.7</td>
<td>16,470</td>
<td>272.3</td>
</tr>
<tr>
<td>Oversides</td>
<td>776,964</td>
<td>-5.086</td>
<td>62.7</td>
<td>107.7</td>
<td>16,238</td>
<td>272.3</td>
</tr>
<tr>
<td>Pass 2nd</td>
<td>778,815</td>
<td>-2.726</td>
<td>28.1</td>
<td>101.6</td>
<td>16,524</td>
<td>116.1</td>
</tr>
<tr>
<td>Oversides</td>
<td>776,306</td>
<td>-2.515</td>
<td>44.6</td>
<td>101.6</td>
<td>16,559</td>
<td>122.7</td>
</tr>
<tr>
<td>Pass 3rd</td>
<td>776,755</td>
<td>45.5</td>
<td>26.4</td>
<td>105.5</td>
<td>16,619</td>
<td>47.9</td>
</tr>
<tr>
<td>Oversides</td>
<td>774,876</td>
<td>-1.879</td>
<td>27.1</td>
<td>105.5</td>
<td>16,619</td>
<td>47.9</td>
</tr>
<tr>
<td>Total</td>
<td>-51,068</td>
<td></td>
<td>758.9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Total costs in dollars/week
* Percent of flow with travel times over those in base case.
* Average number of times a shipment is handled at a breakbulk, times 100
* Average pounds per 28-foot trailer with 20,000 pound capacity
* CPU seconds on an IBM 3081
* Level I sequence DEB, DBE, DBB, ABB, ABE, AEB.

### Table II

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Savings</th>
<th>Percent Over Base</th>
<th>Transfer Rate</th>
<th>Load Average</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>825,562</td>
<td>4.6</td>
<td>120.7</td>
<td>15,826</td>
<td>92.6</td>
<td></td>
</tr>
<tr>
<td>Oversides</td>
<td>815,566</td>
<td>-10.996</td>
<td>9.6</td>
<td>120.7</td>
<td>16,238</td>
<td>92.6</td>
</tr>
<tr>
<td>Pass 1st</td>
<td>780,896</td>
<td>-44.846</td>
<td>21.4</td>
<td>108.6</td>
<td>16,612</td>
<td>248.3</td>
</tr>
<tr>
<td>Oversides</td>
<td>779,803</td>
<td>-1.092</td>
<td>60.9</td>
<td>108.6</td>
<td>16,612</td>
<td>248.3</td>
</tr>
<tr>
<td>Pass 2nd</td>
<td>777,703</td>
<td>-2.049</td>
<td>24.3</td>
<td>104.9</td>
<td>16,612</td>
<td>172.5</td>
</tr>
<tr>
<td>Oversides</td>
<td>776,817</td>
<td>-9.965</td>
<td>35.2</td>
<td>104.9</td>
<td>16,612</td>
<td>172.5</td>
</tr>
<tr>
<td>Pass 3rd</td>
<td>777,780</td>
<td>96.3</td>
<td>24.6</td>
<td>104.6</td>
<td>16,619</td>
<td>124.1</td>
</tr>
<tr>
<td>Oversides</td>
<td>775,351</td>
<td>-2.429</td>
<td>25.2</td>
<td>104.2</td>
<td>16,619</td>
<td>124.1</td>
</tr>
<tr>
<td>Total</td>
<td>-51,011</td>
<td></td>
<td>783.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Level I sequence AEB, ABE, ABB, DDB, DBE, DEB.
at a much lower computational burden than optimizing them during the network design process. However, while the difference is significant in the context of tactical planning, it does not appear to be justified in strategic planning applications, such as studies to locate new breakbulks.

It is interesting to observe from these runs the rate of convergence of the procedures. Tables I and II indicate that the majority of the improvements is found in the first pass, while only a few additional savings are found in the second pass. The third pass in all three cases found negligible improvements. In Table I, passes 2 and 3 actually produced a net increase in costs before the overrides were reoptimized. As mentioned earlier, this can occur because when a link is added, followed by an attempt to increase flow on the added link, and then is dropped, the overrides are not reset to their original values. Such errors are generally corrected after the overrides are reoptimized.

The last question that was addressed was the importance of the level II search sequence. The rule of using links with the smallest flow to sequence the dropping of links was demonstrated by Ref. [4] to be an effective approach, and this finding still held true in this research. The more difficult and interesting problem is sequencing the order in which links are considered for addition. The logic described in the previous section attempted to quickly estimate cost savings from a potential link addition, and then sorted links on the basis of this estimated savings. To compare the effectiveness of this approach, two other ideas were also tested. The first used the concept of local flow on a link. To understand the concept of local flow, it is necessary to introduce the relationship between certain end of lines and breakbulks. Each end of line was associated with a primary or "mother" breakbulk which is determined by the carrier and provided as an input. If a breakbulk is the primary breakbulk for a group of end of lines, those end of lines are referred to as the satellites of that breakbulk. If direct service is offered from breakbulk A to breakbulk B, the local flow on that link consists of flow originating at A and its satellites and terminating at

![Fig. 4. Comparison of effectiveness of search for link additions using estimated cost savings, local flow and random numbers.](image-url)
the very low percentage of good additions that exist in the entire population.

7. SUMMARY AND DIRECTIONS FOR FURTHER RESEARCH

This research has summarized one approach for solving the load planning problem of LTL motor carrier networks. The strength of the approach lies in the fact that it explicitly recognizes the structure of most LTL networks and uses this structure both to improve the efficiency of the algorithms as well as to sequence the search. The numerical experiments that have been conducted to date suggest the procedure performs a good job of identifying a local minimum, and that this solution appears to be relatively robust with respect to how the search is conducted. Despite this apparent success, several important research issues remain which can be divided into the following general areas:

- **The routing subproblem**—Reoptimizing LCPO at times attracted too much flow to a link added to the network, a problem not corrected by the current logic. The problem is that too much flow is subtracted off other links in the network. Improved procedures are needed for reoptimizing RSP in a more precise manner.

- **More complex local improvements**—Local improvements which simultaneously test several link additions/deletions needed to be designed. Such combined changes can be based on identifying the most likely interactions in the network.

- **A theoretical lower bound**—It is simply not possible at this time to rigorously evaluate the success of the local improvement without a tight lower bound on the objective function.

- **Level of service**—Improved methods are required for guaranteeing that actual total travel time constraints are met, or that the total flow over standard is reduced. Such constraints are difficult to handle efficiently due to their large number and the fact that they require tracing paths to determine total travel times.

ACKNOWLEDGMENTS

This research is part of a larger project funded by IU International, Inc., and by a Presidential Young Investigator Award, grant ECE-8451466 from the National Science Foundation. The project has involved the design and implementation of APOLLO, an interactive load planning system that was designed jointly with Yosef Sheffi at MIT. Some of the software for manipulating overrides was written by Ioannis Koskosidis. Especially appreciated has been the valuable feedback provided by the Industrial Engineering Department at Ryde/PIE which is currently using APOLLO, in particular Paul Bodenstein, Bob Stewart, Mark Johnson, Ralph White, Steve Hassler and Bob Radford. The paper has also benefited significantly from careful readings by the referees and the associate editor.

REFERENCES


