Optimization of Group Line-Haul Operations for Motor Carriers Using Twin Trailers

JONATHAN ECKSTEIN AND YOSEF SHEFFI

Group line-haul operation involves the daily movement of trailers between a central breakbulk terminal and a set of end-of-line satellite terminals. When each tractor can pull two trailers, there are many possibilities for creating tractor tours that accomplish the required pickup, delivery, and empty-balancing operations. The challenge is to create optimal tours that minimize transportation costs. An optimization procedure is described that is based on a branch-and-bound framework. It involves Lagrangian relaxation for lower bounds and two upper-bound heuristics for solving this problem.

Less-than-truckload (LTL) motor carriers transport shipments (mostly ranging between a few hundred and a few thousand pounds) between many origins and destinations. Typically, large LTL carriers maintain a three-tiered network that operates as follows:

1. Shipments are picked up from individual customers and taken a short distance to a local end-of-line (EOL), or city, terminal.
2. Local shipments are consolidated in the EOL terminal and transferred to a regional breakbulk terminal, or "break," up to a few hundred miles away. There the incoming shipments are sorted and consolidated by destination into outbound trailers.
3. Each shipment then travels over a network of main-line routes (interconnecting breakbulk terminals) until it reaches the break serving its destination region.
4. In the reverse of Step 2, each shipment travels to the EOL terminal serving its destination city.
5. In the reverse of Step 1, shipments are delivered to individual consignees.

The focus of this paper is on the use of twin trailer trucks in the second step, known as group line-haul operations because LTL networks can be divided into groups, each containing a break and a set of EOL terminals connected to it.

Over the past few years, regulatory changes have allowed twin trailer trucks to be used much more widely on U.S. highways. In such combinations, a single tractor may haul two 28-ft trailers ("doubles" or "pups") instead of one 45- or 48-ft trailer (a van or "semi"). A tractor may also travel alone ("bobtail") or pull just one short trailer. Doubles have proved extremely popular with LTL carriers, particularly on main-line routes [see report by Sheffi and Powell *(1)*]. There they provide extra carrying capacity and improve the level of service without increasing costs.

Although doubles cannot be used in city pickup-and-delivery operations, they have the potential for reducing the costs of group line-haul operations between the EOL terminals and the regional break. The means of achieving such savings are not always obvious because of the combinatorial nature of the problem.

An optimization procedure is outlined for determining daily routes for twin trailer combinations in group line-haul operations. It is based on a multicommodity aggregate flow formulation [see paper by Magnanti *(2)* for classification of methods]. The solution method combines a Lagrangian relaxation for calculating lower bounds with two incumbent generation heuristics to calculate upper bounds, all imbedded in a branch-and-bound (B&B) framework. First the problem is presented in detail; then it is formulated as an integer program. The Lagrangian relaxation is discussed, and the heuristics for the upper bound are described. The details of the B&B procedure are outlined, and, last, some numerical results and a concluding section are presented.

PROBLEM STATEMENT

A group line-haul operation includes one break and a set (5 to 50) of EOL terminals; the latter are responsible for delivery and pickup of shipments at customer locations during business hours. Consequently, the group line-haul operation takes place, for the most part, during the night.

Each afternoon the line-haul planner, at the break, receives data on the number of trailers to be picked up from and delivered to each EOL terminal. Because the operation is repeated nightly, the planner must also consider trailer balancing—empty trailers should be removed from terminals when deliveries exceed pickups and supplied to terminals when pickups exceed deliveries. [In some cases (not considered here) empty trailers are balanced only on a weekly basis. In these cases the trailer inventory at each EOL terminal on each day is a decision variable.]

Trailer movements are accomplished by tractors that can pull up to two trailers at a time. At each terminal a tractor may drop off or pick up either one or two trailers (or do both). The tractor pool must also be balanced daily.

The objective is to cover all the required trailer movements with the minimum number of total tractor miles. The distances between all terminals (naturally) obey the triangle inequality.

Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

These distances are typically small enough (and the supply of tractors is large enough) so that the minimum mileage solution can be executed in one day.

Consider a group line-haul operation including one break and a set of EOL terminals numbered 1, 2, . . ., n. Let p_i denote the number of trailers to be picked up on a given day at EOL terminal i and let d_i denote the number of trailers to be delivered at i.

To get a feel for the options available to the planner, consider two EOL terminals (1 and 2), each requiring delivery and pickup of a single trailer (i.e., $p_1 = d_1 = p_2 = d_2 = 1$). If a tractor could haul only one trailer at a time, the solution would be simple. One tractor pulling outbound freight would be dispatched from the break to each EOL terminal. Each of these two tractors would then drop off its outbound load and pick up an inbound load to take back to the break (Figure 1).

If a tractor can haul two trailers, however, there is a lower-mileage solution using one tractor tour to visit both EOL terminals. The tractor is dispatched from the break pulling both outbound trailers. At the first EOL terminal, it exchanges one of these trailers for a trailer loaded with inbound shipments and then proceeds to the second EOL terminal. There it exchanges the second outbound trailer for an inbound one and then continues to the break. This solution is shown in Figure 2. By the triangle inequality, there are fewer tractor miles in the second solution than in the first.

Although this last example could be solved by inspection, larger problems are dramatically more complex. To see this, consider the four-EOL example shown in Figure 3.

One of the possible solutions resulting in minimum tractor mileage for these data is shown in Figure 4, in which each trailer is represented by a box labeled with the destination of its contents, where B stands for the break and empty trailers are unlabeled. Each set of two adjacent boxes represents a twin-trailer combination pulled by a tractor-trailer. Clearly, the combinatorics of the problem can get quite involved, even for a case with so few nodes.

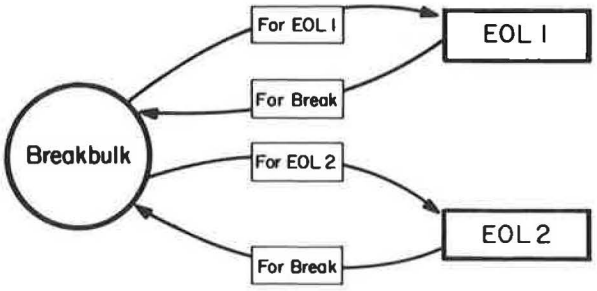

FIGURE 1 Group line-haul operation with two EOL terminals and one trailer per truck.

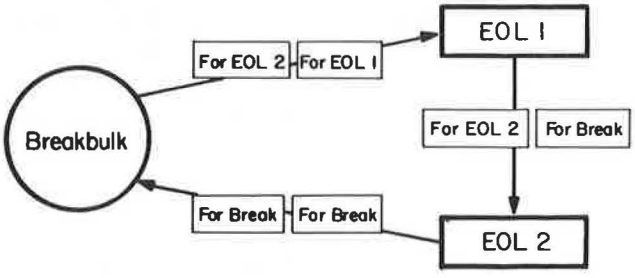

FIGURE 2 Group line-haul operation with two trailers per truck.

The group line-haul problem addressed here exhibits many elements of classical vehicle routing problems: the objective is to create tractor tours from a depot (the break) visiting all customers (EOL terminals) and picking up and dropping off shipments (trailers). A comprehensive review of vehicle routing and practice has been given by Bodin et al. *(3)*. Note, however, that the group line-haul problem addressed here differs from that body of literature in several important ways: (a) shipments (number of inbound and outbound trailers) are often larger than the vehicle size (trailer-pulling capacity), and (b) both pickups and deliveries as well as empty balancing are involved, rather than a single operation.

A better background for the work reported here is provided by Magnanti's survey *(2)*. The emphasis there is on formulations and mathematical programming issues related to basic vehicle-routing problems.

A related and somewhat more general problem is railroad blocking. There the question is which set of freight cars should make up blocks on particular trains. This is a generalization of the group line-haul problem because locomotives can carry more than two cars and because cars move between multiple origin and destination yards (rather than in and out of a central terminal). Assad *(4)* explores several methods for dealing with that problem.

FORMULATION

Consider a terminal group with one break, numbered 0, and n EOL terminals (numbered 1, . . ., n as before) with pickups and deliveries p_i and d_i, respectively, at each i. Define

$$d_0 \equiv -\sum_{i=1}^{n} d_i \qquad p_0 \equiv -\sum_{i=1}^{n} p_i \tag{1}$$

and

$$\mathbf{p} \equiv (p_0, \ldots, p_n)^T$$
$$\mathbf{d} \equiv (d_0, \ldots, d_n)^T \tag{2}$$

Further, define a complete network whose nodes are these $n + 1$ terminals. In this network let $\mathbf{A}$ be the node-arc incidence matrix and $\mathbf{c}$ be the vector of corresponding distances or costs (c_{ij}) of driving a tractor from node i to node j.

Now define four different flow vectors $\mathbf{t}$, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{e}$ over the network given by $\mathbf{A}$. These vectors are all conformal to $\mathbf{c}$:

$\mathbf{t}$ is a vector of scalers t_{ij}, each giving the number of tractors on arc (i, j);

$\mathbf{x}$ is a vector of scalers x_{ij}, each giving the number of outbound trailers (from the break to the EOL terminals) on arc (i, j);

$\mathbf{y}$ is a vector of scalers y_{ij}, each giving the number of inbound trailers (to the break) on link (i, j); and

$\mathbf{e}$ is a vector of scalers e_{ij}, each giving the number of empty trailers moving on link (i, j).

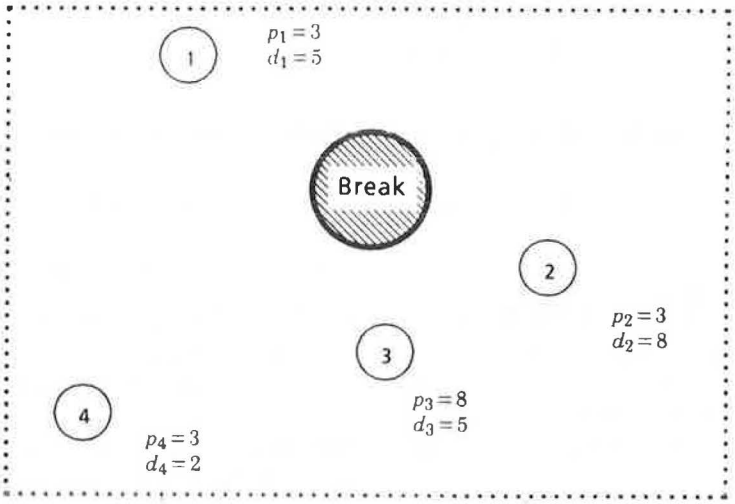

FIGURE 3 More complex example: four EOL terminals.

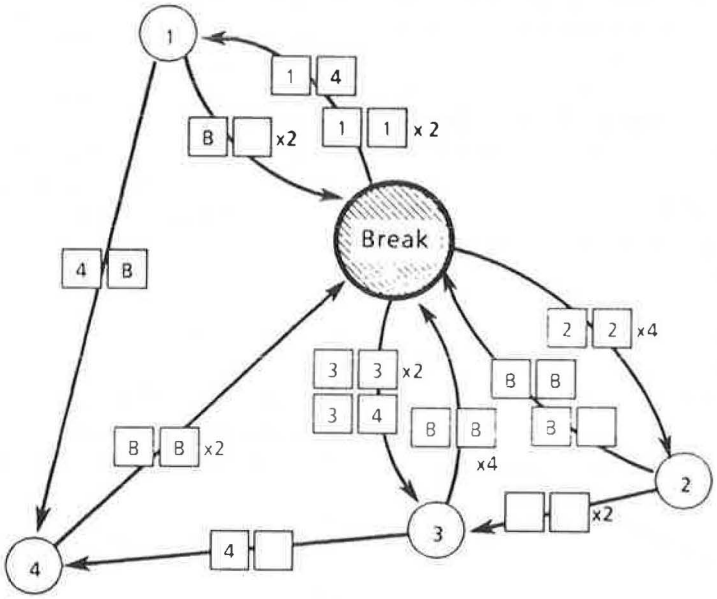

FIGURE 4 Solution for example in Figure 3.

The group line-haul problem can now be formulated as follows:

Minimize

$$\mathbf{c}^T\mathbf{t} \tag{3a}$$

such that

$$\mathbf{At} = 0 \tag{3b}$$

$$\mathbf{Ax} = -\mathbf{d} \tag{3c}$$

$$\mathbf{Ay} = \mathbf{p} \tag{3d}$$

$$\mathbf{Ae} = \mathbf{d} - \mathbf{p} \tag{3e}$$

$$2\mathbf{t} - \mathbf{x} - \mathbf{y} - \mathbf{e} \geq 0 \tag{3f}$$

$$\mathbf{t}, \mathbf{x}, \mathbf{y}, \mathbf{e} \geq 0 \tag{3g}$$

$$\mathbf{t}, \mathbf{x}, \mathbf{y}, \mathbf{e} \quad \text{integer} \tag{3h}$$

Constraint 3b guarantees the flow conservation for tractors, and constraints 3c and 3d guarantee that all pickups and deliveries are made. If $p_i - d_i > 0$, that number of empty trailers has to be supplied to terminal i, whereas if $p_i - d_i < 0$, then $|p_i - d_i|$ trailers must be removed from i. Thus Equation 3e requires the trailer inventory to be stationary. Constraint 3f couples the tractor to the trailers. It says that a tractor can pull at most two trailers; that is,

$$1/2(x_{ij} + y_{ij} + e_{ij}) \leq t_{ij} \qquad \forall (i, j) \tag{4}$$

Equation 3f is a rearrangement of Equation 4 in vector form.

Note that the optimum value of $\mathbf{t}$ is a flow vector and not an explicit set of driver instructions. [This formulation resembles

the one given by Gavish and Graves *(5)* for the traveling salesman problem.] To extract such instructions, the solution has to be decomposed into tours (typically all beginning and ending at the same terminal) and the trailers have to be assigned to the legs of all such tours. Such an extraction is clearly possible: because **t** is a balanced flow with no exogenous sources or sinks, it can be decomposed into tours by a process very similar to that for finding a Euler tour of an even-degree graph.

The solution method used here is based on a B&B method. The lower bound at every B&B subproblem is derived from a Lagrangian relaxation, whereas the incumbent generation is based on ad hoc heuristics. The details of the B&B implementation are explained two sections later, after the relaxation has been outlined and the heuristics have been explained.

THE LAGRANGIAN DUAL

Program 3 consists of four separate network problems linked by additional constraints. A standard technique for dealing with problems with such recognizable embedded structure is to dualize the linking constraints *(6, 7)*. This yields, for each nonnegative $\mathbf{u} \in \Re^{(n+1)n}$ [there are $(n+1)n$ linking constraints], the following Lagrangian relaxation:

$$L(\mathbf{u}) = \text{Min } \mathbf{c}^T\mathbf{t} - \mathbf{u}^T(2\mathbf{t} - \mathbf{x} - \mathbf{y} - \mathbf{e}) \tag{5a}$$

such that

$$\mathbf{At} = 0 \tag{5b}$$

$$\mathbf{Ax} = -\mathbf{d} \tag{5c}$$

$$\mathbf{Ay} = \mathbf{p} \tag{5d}$$

$$\mathbf{Ae} = \mathbf{d} - \mathbf{p} \tag{5e}$$

$$\mathbf{t, x, y, e} \geq 0 \tag{5f}$$

$$\mathbf{t, x, y, e} \quad \text{integer} \tag{5g}$$

$L(\mathbf{u})$ in Equation 5a is a lower bound on the optimum objective value for the original problem for all $\mathbf{u} \geq 0$. The cost coefficients in Equation 5a can then be rearranged to emphasize the structure of four independent network problems. Furthermore, because network problems have integer optima if their right-hand sides are integer, constraint 5g can be dropped from the formulation, which becomes

$$L(\mathbf{u}) = \text{Min } (\mathbf{c} - 2\mathbf{u})^T\mathbf{t} + \mathbf{u}^T\mathbf{x} + \mathbf{u}^T\mathbf{y} + \mathbf{u}^T\mathbf{e} \tag{6a}$$

such that

$$\mathbf{At} = 0 \tag{6b}$$

$$\mathbf{Ax} = -\mathbf{d} \tag{6c}$$

$$\mathbf{Ay} = \mathbf{p} \tag{6d}$$

$$\mathbf{Ae} = \mathbf{d} - \mathbf{p} \tag{6e}$$

$$\mathbf{t, x, y, e} \geq 0 \tag{6f}$$

The best lower bound (D) would be obtained as

$$D \equiv \text{Max}_{\mathbf{u}\geq 0} L(u) \tag{7}$$

By the strong duality theory of linear programming, $D = Z_{LP}$, where Z_{LP} is the optimum of the LP relaxation of program 3. Thus, the Lagrangian relaxation can produce a lower bound to program 3 that is only as tight as the LP relaxation of program 3 (given the correct choice of **u**).

As it turns out, it is possible to pick a priori the values of the multipliers **u** that yield the largest possible value of $L(\mathbf{u})$. The appropriate choice is

$$\mathbf{u}^* \equiv (1/2)\,\mathbf{c} \tag{8}$$

This gives each arc (i, j) an imputed tractor cost of 0 and an imputed trailer cost of $c_{ij}/2$. Essentially, one can think of this as attaching half a tractor to each trailer. The constraint $2\mathbf{t} - \mathbf{x} - \mathbf{y} - \mathbf{e} \geq 0$ is then automatically satisfied with zero slack, and trailers circulate in a shortest-path manner subject to a lowest-cost repositioning of empties.

In order to strengthen relaxation 6, consider the original integer program (3). Note that if d_i trailers must be delivered at EOL terminal i, at least $\lceil d_i/2 \rceil$ tractors must visit that terminal (where $\lceil \cdot \rceil$ denotes the upwards integer rounding function). Similarly, at least $\lceil d_0/2 \rceil$ tractor trips must be dispatched from the break. An analogous argument applies to pickups, and so one can conclude that the minimum number of tractors to pass through terminal i is given by

$$v_i \equiv \text{Max } \{\lceil |p_i|/2 \rceil, \lceil |d_i|/2 \rceil\} \quad \text{for } i = 0, 1, \ldots, n \tag{9}$$

This observation can be used to add the following set of $(n+1)$ constraints to program 3:

$$\sum_{j \neq i} t_{ij} \geq v_i \quad \text{for } i = 0, \ldots, n \tag{10}$$

Such "node-activity" constraints are redundant and may be added to the basic aggregate flow formulation without altering it. However, when the linking constraints are dualized, the node-activity constraints are no longer redundant: they alter the tractor part of the computation of $L(\mathbf{u})$ considerably. With the addition of these constraints this part of the problem becomes

$$\text{Min } (\mathbf{c} - 2\mathbf{u})^T\mathbf{t} \tag{11a}$$

such that

$$\mathbf{At} = 0 \tag{11b}$$

$$\sum_{j \neq i} t_{ij} \geq v_i \qquad \text{for } i = 0, \ldots, n \tag{11c}$$

$$\mathbf{t} \geq 0 \tag{11d}$$

Constraint 11c can be viewed as cutting planes that disallow the previous LP optimum, $\mathbf{t}^* = 1/2(\mathbf{x}^* + \mathbf{y}^* + \mathbf{e}^*)$, unless it is integer (and hence optimal). The Lagrangian relaxation is thus strengthened by the addition of these constraints. By using a

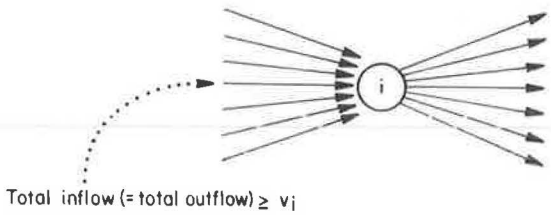

FIGURE 5 Before node splitting.

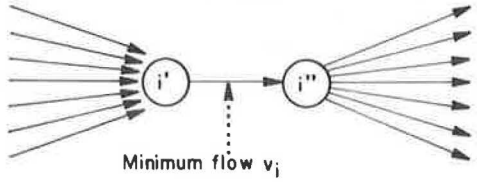

FIGURE 6 After node splitting.

standard "node-splitting" technique *(8)*, constraints 11c may be handled without destroying the network structure of the tractor subproblem (Figures 5 and 6). In practice, the addition of the node-activity constraints improves the efficiency of the Lagrangian B&B method by a factor of approximately 5.

CALCULATING THE LOWER BOUND

At any given point in the B&B tree, Lagrangian 5 (along with constraints 11c and additional separation constraints) has to be solved. Note that the separation constraints include only upper and lower bounds on arc flows, and consequently they do not disturb the network structure of this program. Given $\mathbf{u}$, $L(\mathbf{u})$ can be computed by applying the primal network simplex four times (with the appropriate network modification for the tractor part to enforce the node-activity constraints).

Given a solution to the program consisting of Equations 5 plus constraints 11c plus separation constraints, $L(\mathbf{u})$ is a lower bound to the program consisting of Equations 3 plus separation constraints. To get a set of multipliers that would give a better lower bound, one can use a subgradient method. With this method the new value of the multipliers is

$$\mathbf{u} := \mathbf{u} + s\mathbf{g}, \tag{12a}$$

where the direction $\mathbf{g}$ is given by the subgradient

$$\mathbf{g} \equiv \mathbf{x} + \mathbf{y} + \mathbf{e} - 2\mathbf{t}, \tag{12b}$$

and the step s is given by

$$s = a\frac{Z_{INC} - L(\mathbf{u})}{||\mathbf{g}||^2} \tag{12c}$$

This is a standard subgradient step, where a is the step size and the "target" value of $L(\mathbf{u})$ is taken to be Z_{INC} (the objective function value of the incumbent). This is the highest possible value one may choose for the target. It is reasonable only because the incumbents are usually very good.

The initial value of $\mathbf{u}$ in a given subproblem of the B&B tree is taken to be the terminal $\mathbf{u}$ in the predecessor ("parent") subproblem. In the initial problem, best results were obtained with $\mathbf{u} = \mathbf{c}/4$. This is the center of the "box" $\{\mathbf{u}|0 \le \mathbf{u} \le \mathbf{c}/2\}$ which, as shown in the following, approximates the dual feasible region.

If the convergence of the subgradient algorithm for a given subproblem is slow, it may be best to separate the subproblem. The rule used to terminate is the following:

If for iteration $m > K$,

$$\frac{L(\mathbf{u}) - Z_{INC}}{L(\mathbf{u}_0) - Z_{INC}} \ge 1 - m\delta, \text{ STOP} \tag{13}$$

where $\mathbf{u}_0$ is the first multiplier vector used for the subproblem, and K and δ are adjustable parameters. In other words, the algorithm has to go through at least K iterations. It then splits if the average improvement is no more than a fraction δ of the way to the incumbent value per step.

The reason that a minimum of K iterations must be performed before separation is that one cannot rely upon $L(\mathbf{u})$ to increase at every step, even when it is far from its maximum value for the subproblem at hand. Without the $m > K$ restriction, a single "bad" step at the outset of subproblem analysis would cause an immediate, unnecessary separation. Every such mishap would double the work the method must do to fathom a particular branch of the enumeration tree. Separations are thus costly and should be avoided unless they are absolutely necessary. On small test problems, values of $K = 3$ and $\delta = 0.02$ proved efficient.

Note that if at some iteration, as the result of a subgradient step, $u_{ij} > c_{ij}/2$ for some (i, j), the imputed cost $c_{ij} - 2u_{ij}$ in the Lagrangian tractor network will be negative. Because this entails a high risk of creating a negative cycle in the imputed tractor costs, $\mathbf{u}$ should be restricted to the dual feasible region:

$$U \equiv \{\mathbf{u} \ge 0 | \mathbf{c} - 2\mathbf{u} \text{ has no negative cycles}\} \tag{14}$$

This region is a polytype, but it has an exponentially growing number of constraints. It is therefore easier to make the approximate restriction

$$0 \le \mathbf{u} \le \mathbf{c}/2 \tag{15}$$

which assures, more strongly, that $\mathbf{c} - 2\mathbf{u}$ has no negative cost arcs. After each subgradient step, the resulting new value of $\mathbf{u}$ is projected onto this subset of the feasible region. Because this region is box-shaped, the projection is simply

$$u_{ij} := \text{Max}\{0, \text{Min}\{u_{ij}, c_{ij}/2\}\} \qquad \forall(i, j) \tag{16}$$

Strategies other than projection for enforcing that $0 \le \mathbf{u} \le \mathbf{c}/2$ (such as truncating the step) appear not to allow $L(\mathbf{u})$ to grow as quickly.

INCUMBENT GENERATION

Given a solution to a relaxed subproblem, one needs to find an incumbent. With the Lagrangian-based lower bound, one ob-

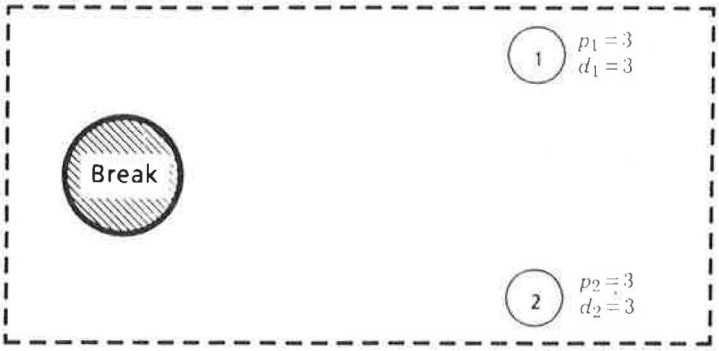

FIGURE 7 Roundup fails—problem data.

vious strategy presents itself. Given the (**t**, **x**, **y**, **e**) optimal in $L(\mathbf{u})$, **t** can be replaced with the minimum-cost integer tractor flow vector $\mathbf{t}_z$ required to support the trailer flows **x**, **y**, and **e**. On each link (i, j), this flow must be at least $(x_{ij} + y_{ij} + e_{ij})/2$, but also integer, hence at least $\lceil (x_{ij} + y_{ij} + e_{ij})/2 \rceil$. However, the tractor flow must also be balanced and consequently t_z should be the optimal solution to the problem

$$\text{Min } \mathbf{c}^T\mathbf{t} \tag{17a}$$

such that

$$\mathbf{At} = 0 \tag{17b}$$

$$\mathbf{t} \geq \lceil 1/2\ (\mathbf{x} + \mathbf{y} + \mathbf{e}) \rceil \tag{17c}$$

Program 17 is a network circulation problem and may be solved by network simplex. The quadruplet $(\mathbf{t}_z, \mathbf{x}, \mathbf{y}, \mathbf{e})$ is then a feasible solution to the integer program 3. Unfortunately, this simple roundup incumbent generation strategy is inadequate. To demonstrate this, consider the simple example problem in Figure 7. One of the two optimal solutions to this group line-haul problem is given in Figure 8. (In the other optimum, the roles of EOL terminals 1 and 2 are interchanged.) Because the algorithm separates only on the t_{ij} (tractor) variables, as $L(\mathbf{u})$ is computed, all trailers will still always be free to take the shortest path to their final destinations. For instance, all outbound traffic for Terminal 2 could take the route shown in Figure 9 or, with a different **u**, the route shown in Figure 10. However, the IP optimum includes routings in which two different paths are used, such as the one shown in Figure 11.

Splitting only on the **t** variables, the roundup heuristic never changes any trailer routes, so it can never discover the optimum. Two courses of action can be used to detect the optimum: separating on trailer variables and creating an incumbent finder that intelligently alters trailer routes. The former

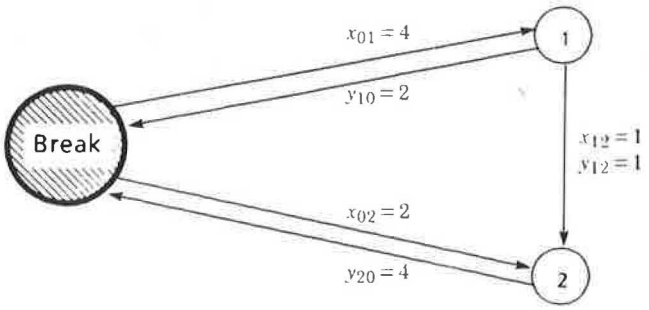

FIGURE 8 Roundup fails—integer optimum.

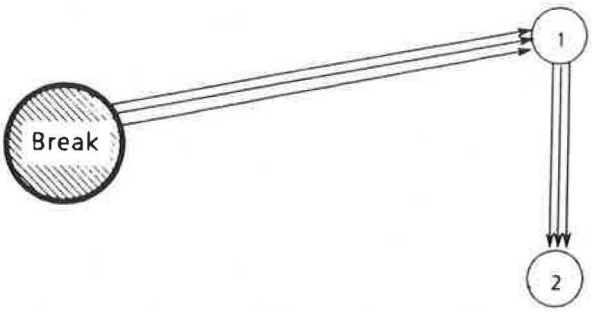

FIGURE 10 Roundup fails—another possible routing from break to 2 in $L(\mathbf{u})$.

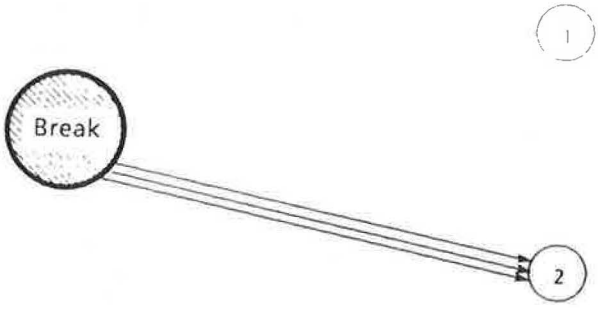

FIGURE 9 Roundup fails—one possible routing from break to 2 in $L(\mathbf{u})$.

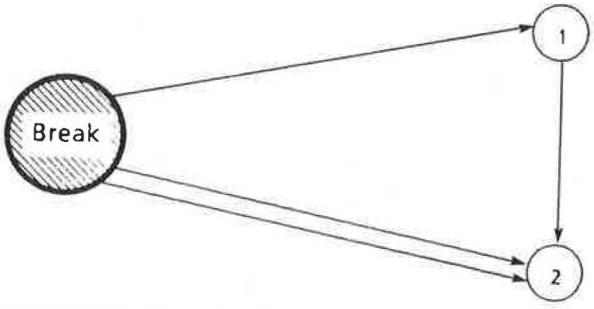

FIGURE 11 Roundup fails—a routing that cannot occur in $L(\mathbf{u})$.

strategy puts most of the burden of finding a solution on the enumeration component of the algorithm—which may lead to a very large B&B tree. Consequently the second course of action was chosen.

In the following sections two incumbent generation schemes based on local improvement heuristics are described. The focus in the first one is on developing a method that will run quickly, whereas the focus of the second is on getting accurate solutions. These methods are explained separately.

Method 1: Simple Local Improvements

The first heuristic takes the $L(\mathbf{u})$ solution and makes some local modifications to it. The trailer routings are slightly perturbed [ignoring the $\mathbf{t}$ part of the $L(\mathbf{u})$ solution] so that the minimum integer tractor movements needed to "cover" them can be reduced. In the combined $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{e}$ solution to $L(\mathbf{u})$, the method detects all patterns similar to those shown in Figure 12, where solid arrows represent arcs with an odd total number of trailers.

The local improvements shown in Figures 13–15 are applied, respectively, to each of the three patterns in Figure 12 (dashed lines represent arcs whose trailer flows have been increased from odd to even values, and boxes represent particular trailers). The modifications are then ranked by their total savings, that is, c_{0b} in the first case, c_{a0} in the second, and

$$(c_{0a} + c_{a0} + c_{0b} + c_{b0}) - (c_{0a} + c_{b0} + c_{ab}) = c_{a0} + c_{0b} - c_{ab} \geq 0 \qquad (18)$$

in the third. These improvements are then implemented in a greedy manner, highest savings first, until no more can be implemented. (Note that some of the detected patterns might overlap so that performing one might preclude applying others.) After these modifications have been made, the flow-based roundup procedure (program 17) is performed to compute the $\mathbf{t}$ part of the candidate incumbent, this time using the modified trailer flows as input.

As shown by Eckstein *(9)* the method described earlier cannot identify all the possible patterns that can be improved to generate a lower-cost tractor flow. More improvement patterns can be found if the minimum tractor covering is calculated first (given a solution of the Lagrangian relaxation) and then the resulting slack capacity is investigated for promising patterns. This is the basis of the second method for generating incumbents.

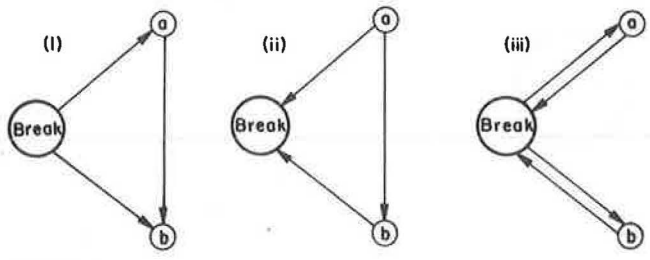

FIGURE 12 Patterns recognized by simple local improvement heuristic.

Method 2: Cycle Splicing

Given a "round-up solution" $(\mathbf{t}_z, \mathbf{x}, \mathbf{y}, \mathbf{e})$, let the slack vector $\mathbf{s}$ be defined as

$$\mathbf{s} \equiv 2\mathbf{t}_z - \mathbf{x} - \mathbf{y} - \mathbf{e} \qquad (19)$$

Since $\mathbf{A}\cdot\mathbf{t} = 0$ (see program 11) and $\mathbf{s} \geq 0$, $\mathbf{a}\cdot\mathbf{s} = 0$. To see this, note the following:

$$\begin{aligned} \mathbf{As} &= 2\mathbf{At}_z - \mathbf{Ax} - \mathbf{Ay} - \mathbf{Ae} \\ &= 0 + \mathbf{d} - \mathbf{p} - (\mathbf{d} - \mathbf{p}) \\ &= 0 \end{aligned} \qquad (20)$$

Thus, the vector of slack capacities $\mathbf{s}$ may also be considered a balanced "flow" in the interterminal network. Now any balanced flow can be expressed as a sum of flows around simple directed cycles. (A simple cycle is defined as one that repeats neither arcs nor nodes.)

Definition: A cycle decomposition of some balanced flow $\mathbf{w}$ ($\mathbf{w} \geq 0$, $\mathbf{Aw} = 0$) is a sequence $H_1, \ldots, H_J$ of (not necessarily distinct) simple directed cycles such that

$$\sum_{l=1}^{J} \mathbf{f}(H_l) = \mathbf{w} \qquad (21)$$

where $\mathbf{f}(H_l)$ denotes a flow vector that has a 1 in each position (i, j) corresponding to an arc of H_l, and zeroes in all other positions.

Note that (if $\mathbf{c}$ has no zero-cost cycles), $\mathbf{s}$ will always decompose into a sequence of distinct simple cycles. Moreover, if $\mathbf{u}$ is strictly triangular (which it is when $\mathbf{u}$ is proportional to $\mathbf{c}$), this decomposition is unique.

Consider two simple cycles, R and S in the decomposition of $\mathbf{s}$, having the property that they cross only at the break. The two cycles can be combined in a splicing operation that inserts one in a given arc of the other. As an example, consider Figure 16, which shows two such cycles. Cycle S can be inserted into arc (i, j) to create the combined cycle shown in Figure 17 to create a substantial cost saving. In this example, the decrease in tractor costs is

$$\Delta Z = c_{0a} + c_{ij} + c_{b0} - c_{ia} - c_{bj} \qquad (22)$$

which, in general, may be positive, negative, or zero.

Note that there are a variety of different splicing opportunities because the roles of R and S may be interchanged, and the insertion arc (i, j) may be varied. However, if the cycles are both short, there will only be a handful of possibilities, some of which will produce savings, whereas others will not.

Note also that this method is applicable to cases in which R and S share some arcs (but there is a slack of at least 2 on all shared arcs). Essentially, each nonshared part of one cycle must be spliced into some nonshared arc of the other, and the number of tractors on all shared arcs is reduced by 1. For example, consider the problem shown in Figure 18. The LP solution (ignoring node-activity constraints) is shown in Figure 19, and the corresponding slack is shown in Figure 20. This slack can be broken down into the cycles S = 0-1-0 and R = 0-2-1-0.

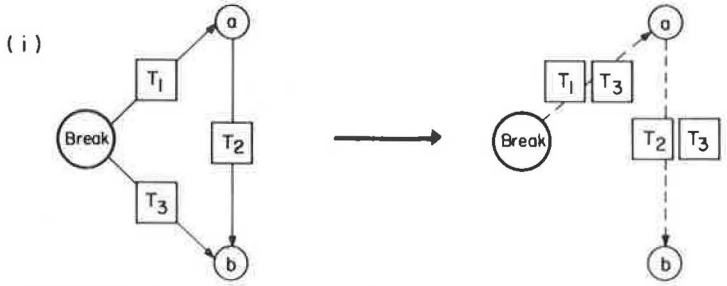

FIGURE 13 Local improvement for pattern (i).

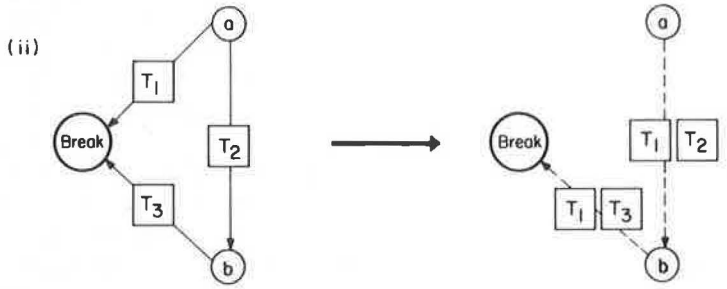

FIGURE 14 Local improvement for pattern (ii).

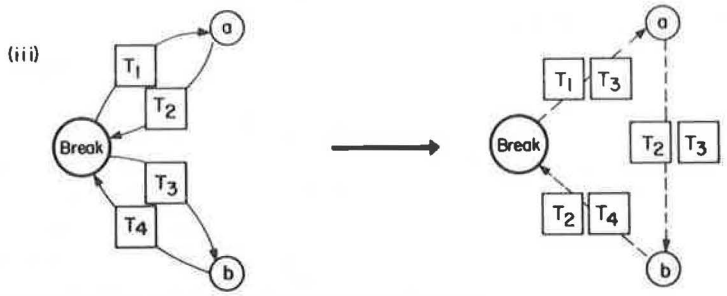

FIGURE 15 Local improvement for pattern (iii).

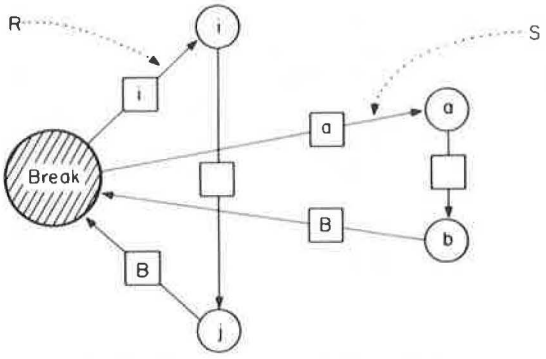

FIGURE 16 Two cycles ready for splicing.

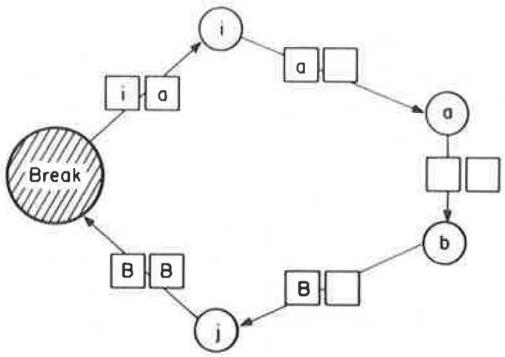

FIGURE 17 Outcome of the splicing operation.

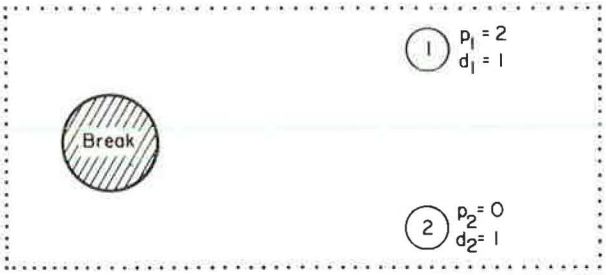

FIGURE 18 Splicing with shared arcs—problem data.

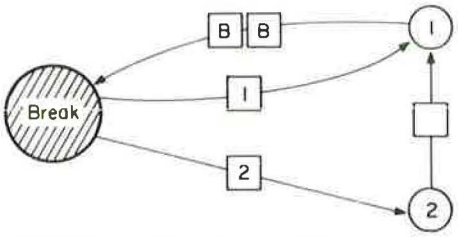

FIGURE 19 Shared-arc splicing—optimum trailer flows.

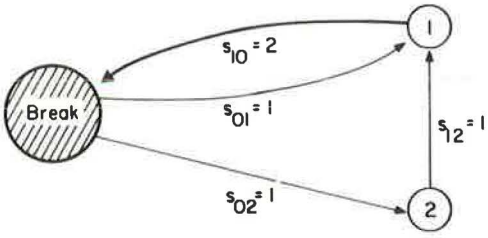

FIGURE 20 Slacks for shared-arc splicing example.

Splicing the nonshared part of R, 0-2-1, into the nonshared arc (0, 1) of S, the IP optimum shown in Figure 21 is obtained.

The cycle-splicing method works as follows:

1. Given **x, y,** and **e** optimal in $L(\mathbf{u})$, it solves the constrained roundup problem (Equations 11) and computes the resulting slack $\mathbf{s}$.
2. It decomposes **s** into a sum of simple cycles.
3. For each pair of such cycles, it examines all the possible ways of splicing them together. For each pair, the heuristic identifies the most attractive splicing opportunity and records it in a list.
4. It performs the recorded splicing operations in a greedy manner, starting with the one with the greatest savings and proceeding to the next most profitable one that is still allowable until the list is exhausted.

Both the simple local improvement heuristic and the cycle-splicing heuristic take as input the **x, y,** and **e** arising from a solution of $L(\mathbf{u})$. They then attempt to juggle some of the routings in order to reduce the cost of covering these trailer movements with tractors. Neither heuristic ever really changes some fundamental aspects of the incoming solution, in particular the assignment of empty trailers. If some EOL terminal i receives a given number of empties from EOL j in the solution

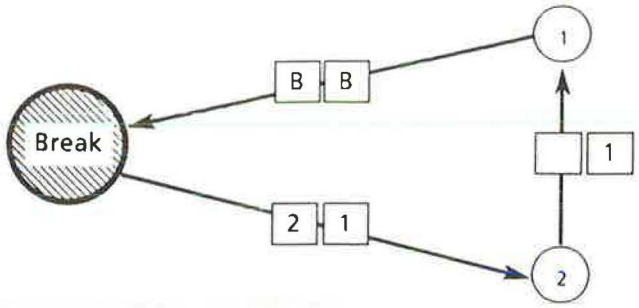

FIGURE 21 Outcome of shared-arc splice (also IP optimum).

to $L(\mathbf{u})$, it will still do so in the perturbed solution output from either heuristic. The only difference will be that certain detours may be inserted in paths taken by these trailers.

In practice, the cycle-splicing heuristic works very well, as shown by Eckstein *(9)*; under certain general conditions the cycles that arise in the decomposition of **s** are always short, allowing for efficient implementation. It is, however, somewhat slow, and implementing it for every subproblem actually slows down the overall run time of the Lagrangian B&B procedure (as compared with using Method 1). A more efficient approach is to use the cycle-splicing heuristic just once, at the very beginning of the algorithm, and the much faster simple local improvement heuristic (Method 1) at every iteration of every subsequent subproblem processed. In this way, one gets the advantage of a good initial incumbent without the computational burden of running the cycle-splicing heuristic repetitively.

LAGRANGIAN B&B PROCEDURE

As mentioned in the first section, the Lagrangian relaxation and the heuristic incumbent generation methods are used within a B&B procedure. Figure 22 shows a general flowchart of the calculation involved in Lagrangian-based B&B methods following Fisher *(6)*. A description has been given of how a **u** vector is chosen for each subproblem (block 2 in Figure 22), how $L(\mathbf{u})$ is calculated for a given **u** (block 3), and how the dual iterations are performed (block 8) and when they are terminated (block 7). The generation of a new incumbent solution (block 5) was described in the previous section. Here, some of the implementation details of the B&B procedure are given.

At each point in the solution procedure the B&B procedure deals with one subproblem (a point on the enumeration tree), that is, the original integer program with some added separation constraints. An active subproblem is one that has not been further separated and is thus an end point of the tree. At any given time, each subproblem q has some best lower bound β_q on its LP-relaxed objective value. This lower bound is the highest value of $L(\mathbf{u})$ found for the subproblem, or any of its ancestors, for all the **u**'s tried so far. The lowest value the global integer optimum could possibly have is

$$\beta \equiv \text{Min } \{\beta_q | q \text{ an active subproblem}\} \tag{23}$$

The B&B method tries to increase β and decrease the upper-

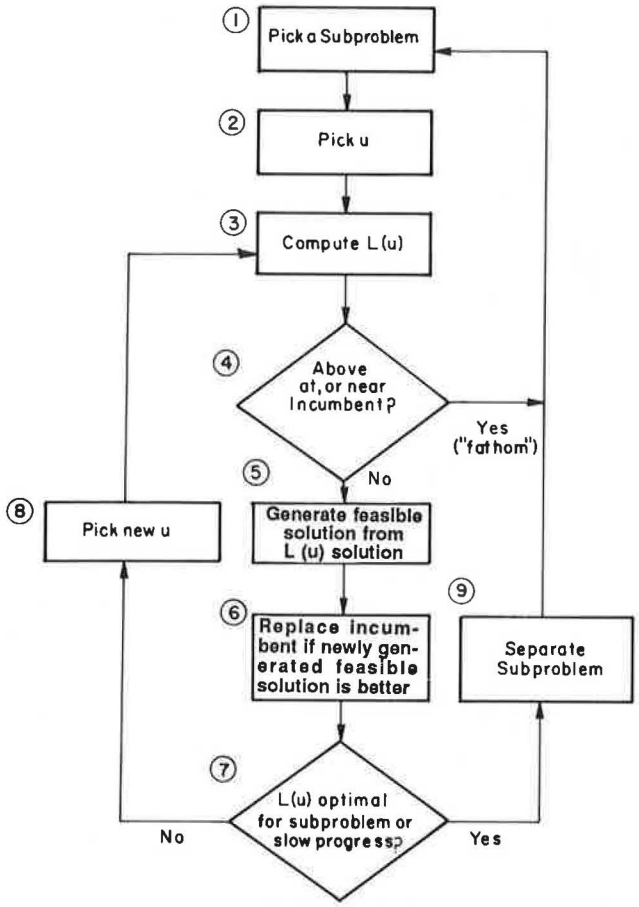

FIGURE 22 General Lagranglan branch and bound.

bound Z_{INC} (the objective value for the incumbent) until they are near enough to conclude that Z_{INC} is acceptably close to optimality.

Fathoming

A subproblem is fathomed (removed from further consideration) if a lower bound for its objective value provided by an $L(\mathbf{u})$ computation is within P percent of the upper bound on the global optimum provided by the incumbent. Such a strategy guarantees that the final solution is within P percent of optimality. The fathoming condition is the following:

$$\lceil L^*\mathbf{u}) \rceil \geq \left(1 - \frac{P}{100}\right) Z_{INC} \tag{24}$$

The user-specified parameter P can be used to explicitly trade off between the accuracy of the solution and the speed of obtaining it.

Separation

If slow improvement is detected in solving the Lagrangian relaxation of a given subproblem, it is split in two. In practice, it seems most efficient to split on the tractor variables t_{ij} rather than on the trailer variables x_{ij}, y_{ij}, or e_{ij}. Each split requires that t_{ij} be less than or equal to some integer m in one offspring subproblem and greater than or equal to $m + 1$ in the other.

Good computational experience was obtained by splitting on the t_{ij} for the longest arc (i, j) for which the number of trailers in the current $L(\mathbf{u})$ solution is odd. Thus the following constraints are added to the offspring subproblem:

$$t_{ij} \leq m \tag{25a}$$

$$t_{ij} \geq m + 1 \tag{25b}$$

The value of m is chosen to be $\lfloor x_{ij} + y_{ij} + e_{ij} \rfloor$ unless t_{ij} has been otherwise constrained ($\lfloor \cdot \rfloor$ denotes the floor or integer round-down function).

The separation scheme used in practice is somewhat more complicated; a "scoring" method is used for choosing the splitting arc. Each arc with odd trailer flow is assigned a score based on its length, whether it emanates from or terminates at the break, and whether it was separated on before. The algorithm then splits on the arc with the highest score. This method provided only about a 10 percent run time improvement over the simpler "longest odd arc" method mentioned before.

For the "lower" ($t_{ij} \leq m$) offspring, where at most m tractors are allowed on some arc (i, j), it is clear that there can be at most $2m$ trailers of each kind on (i, j). Thus the following redundant constraints can be enforced:

$$x_{ij} \leq 2m \tag{26a}$$

$$y_{ij} \leq 2m \tag{26b}$$

$$e_{ij} \leq 2m \tag{26c}$$

These simple upper bounds strengthen the Lagrangian relaxation and can be added without breaking network structure.

Subproblem Selection

After the algorithm has fathomed or separated a subproblem, it is faced with the decision of which subproblem to try next (unless there are none left unfathomed, in which case it terminates). The procedure uses here a simple rule of processing the problem, q, with the lowest βq.

The algorithm also includes a feature that allows it to switch to high-β_p subproblems in the event that there are so many active subproblems that the program is close to exhausting its virtual memory allocation. The intent is that these subproblems may be fathomed relatively quickly, freeing up memory for the more important ones. This feature allows the algorithm to handle larger problems with a given amount of memory, albeit with some speed penalty.

Basis Preservation

To improve run times the procedure uses information from earlier network simplex bases to speed up calls to the network simplex code. When $L(\mathbf{u})$ is computed, four network simplex

optimizations must be performed. At each subgradient iteration, the previous four optimal bases for the subproblem are used as the four starting bases. Because the optimal bases for two consecutive values of **u** should resemble one another, fewer pivots may be needed than if all initial bases are constructed from scratch.

This idea is carried one step further: when $L(\mathbf{u})$ is computed first for a subproblem, the procedure starts essentially with the four bases that were optimal in the last iteration of its parent. The slight difficulty here is that the added separation constraint may make one or more of the old optimal bases infeasible for the offspring. To see how this is handled, assume, for example, that a constraint $t_{ij} \leq m$ was added to a parent problem in which $t_{ij} = r > m$, rendering the parent basis infeasible in the offspring. Now, all the network representations contain a "super transshipment" node connected to all other nodes by "artificial" arcs of very high ("big M") cost. To maintain feasibility in the child, the flows are perturbed as shown in Figure 23.

By also setting the maximum flow capacity of the two artificial arcs to $r - m$, the procedure avoids having to add them to the basis (although either of them could already be in the basis in a degenerate manner), and the original spanning tree of the parent basis remains valid in the subproblem's perturbed network. Of course, when the network simplex routine is called, the artificial arcs will immediately have flow removed from them, because they have such high costs. Analogous techniques can be used for the addition of constraints of the form $t_{ij} \leq m + 1$, and also for the **x**, **y**, and **e** bases.

Old basis information could have been retained without perturbations by using a dual method to reoptimize the offspring of a subproblem, as in standard B&B methods, but this would have required the implementation of both primal and dual network simplex algorithms.

The drawback to using basis-preservation methods is that information must be stored for all active subproblems, increasing program memory requirements. This can be alleviated by using a depth-first tree exploration strategy in which the parent basis for one of the offspring of a subproblem is used without having to allocate any more memory, as long as that one offspring is analyzed immediately after separation.

COMPUTATIONAL PERFORMANCE

The Lagrangian B&B procedure described earlier was implemented on a VAX 11/780 minicomputer using VAX FORTRAN. (The computational power of this machine, at least for compute-bound numerical tasks, is now roughly matched by some of the recently released work stations and "high-end" personal computers.)

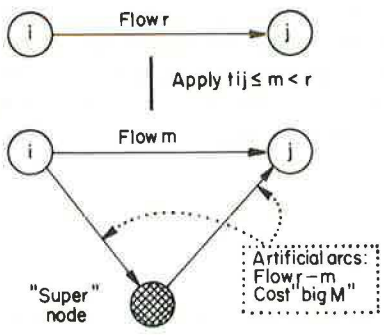

FIGURE 23 Basis modification for a violated upper bound.

The code was run on three sets of 20 randomly generated problems, one set with 4 EOL terminals per instance, one with 8, and one with 16. For these problems, EOL terminals were uniformly distributed over the interstices of a 40-by-40 grid centered on the break, with all p_i's and d_i's drawn from the probability mass function (PMF)

$$p(x) = \begin{cases} 0.4, & x = 1 \\ 0.4, & x = 2 \\ 0.1, & x = 3 \\ 0.1, & x = 4 \end{cases} \tag{27}$$

This PMF resembles values that might arise in practice. Distances were Euclidean, rounded up to integer values, with small adjustments and "stop-off" costs added to ensure that the triangle inequality was always strictly met.

Each of the 4-, 8-, and 16-node problem sets was run at varying levels of the percentage-fathoming parameter P. For budgetary reasons, each individual problem run was limited to roughly 5 megabytes of virtual memory and 15 min of CPU time. In summary, results were excellent for the 4-node problems, acceptable for 8 nodes, and somewhat disappointing for the 16-node examples.

Table 1 gives the results for selected combinations of numbers of EOL terminals and levels of P. It should be noted that runs were simply cut off if the available time or memory limits had to be exceeded. The method is quite memory-intensive, because basis information is retained from subproblem to subproblem. On the 8- and 16-node problems, upon reaching about 300 active subproblems, the algorithm would often switch to the high-β_q mode in which it attempted to fathom unpromising subproblems first in an effort to free up space. This procedure tended to slow down convergence and was sometimes unsuccessful in holding down memory requirements (when high-β_q subproblems could not be fathomed), in which case the program simply halted upon reaching its 5-megabyte storage limit.

As the results given in Table 1 suggest, the B&B procedure can easily solve small problems but does not perform well for large ones. In particular, the 16-node runs produce a duality gap only marginally smaller than that one would get by simply running the cycle-splicing heuristic and the LP relaxation and then comparing the two. None of the twenty 16-node cases run showed any improvement in the incumbent after the first iteration of the first subproblem, and the improvement in the lower bound attributable to enumeration was not very great. Thus, the best course in practice is to use the cycle-splicing heuristic as a stand-alone procedure and to optionally employ the rest of the algorithm as a means of assessing nearness to optimality.

In a detailed analysis of the smaller problems, there appeared to be a strong dependence of run time on the initial value of **u**. On average, the best results were obtained with $\mathbf{u} = \mathbf{c}/4$. However, for individual problems, different values would sometimes work better. This phenomenon, once understood, could perhaps be exploited to improve the updating of **u**.

TABLE 1 PERFORMANCE OF LAGRANGIAN B&B METHOD

No. of EOL Terminals	Fathoming Percentage P	Run Time (sec)		Duality Gap (%)	
		Mean	Median	Mean	Median
4	0	16.7	4.3	0.1	0.0
4	5	5.6	0.9	5.0	5.0
8	0	439.0	348.9	1.6	0.0
8	5	166.7	32.9	5.2	5.0
16	5	900+	900+	15.8	16.5

CONCLUSIONS

A method for optimizing tractor-trailer and twin-trailer movements in a group line-haul operation has been described. The method is based on a B&B procedure in which the lower bounds are calculated by a Lagrangian relaxation. The first incumbent is generated by a cycle-splicing heuristic and the incumbents at every subproblem are generated by a simple local improvement heuristic.

For small problems the B&B procedure worked well, but for larger ones the cycle-splicing heuristic should be used as a stand-alone method. The shortcomings of the B&B procedure may be alleviated by improving the dual step and thus setting the multiplier variables **u** in a more efficient way.

Other solution methods for this problem may be based on extensions of the cycle-splicing heuristics in which a matching problem is solved in order to decide on the best splicing combinations *(9)*. Alternatively one can think of a set covering formulations based on tractor tours *(10, 11)*.

ACKNOWLEDGMENT

Several individuals from Consolidated Freightways, Inc., helped in the understanding of group line-haul operations and the problems involved. They include Lee Frazee, Robert McDonald, Philip Seeley, Robert Blackburn, Toby Asarese, and Al Delara. In addition, Gene Hughs and Tom Meyers of United Parcel Service helped in the understanding of the (somewhat different) UPS twin-trailer matching problem. Thanks are expressed to all these individuals.

REFERENCES

1. Y. Sheffi and W. Powell. *Interactive Optimization for LTL Network Design*. Center for Transportation Studies Report 85-17. Massachusetts Institute of Technology, Cambridge, 1985.
2. T. L. Magnanti. Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects. *Networks*, Vol. 11, 1987, pp. 179–213.
3. L. Bodin et al. Routing and Scheduling of Vehicles and Crews: The State of the Art. *Computers and Operations Research*, Vol. 10, 1983, pp. 63–211.
4. A. A. Assad. *Modeling Rail Freight Management*. Ph.D. thesis. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, 1978.
5. B. Gavish and S. C. Graves. *The Traveling Salesman and Related Problems*. Working Paper OR 078-78. Operations Research Center, Massachusetts Institute of Technology, Cambridge, 1978.
6. M. L. Fisher. The Lagrangian Relaxation for Solving Integer Programming Problems. *Management Science*, Vol. 27, 1981, pp. 1–18.
7. M. L. Fisher. An Applications Oriented Guide to Lagrangian Relaxation. *Interfaces*, Vol. 15, No. 2, 1985, pp. 10–21.
8. B. L. Golden and T. L. Magnanti. Course Notes, Course 15.082. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, 1985.
9. J. Eckstein. *Routing Methods for Twin-Trailer Trucks*. Master's thesis. Operations Research Center, Massachusetts Institute of Technology, Cambridge, 1986.
10. M. L. Balinksi and R. E. Quandt. On an Integer Program for a Delivery Problem. *Operations Research*, Vol. 12, 1964, pp. 300–304.
11. F. H. Cullen, J. J. Travis, and H. D. Ratliff. Set Partitioning Heuristics for Interactive Optimization. *Networks*, Vol. 11, 1981, pp. 125–143.