Pup Matching: Model Formulations and Solution Approaches

J. M. Bossert and T. L. Magnanti

Abstract— **We model Pup Matching, the logistics problem of matching or pairing semitrailers known as pups to cabs able to tow one or two of the pups simultaneously, as an $\mathcal{NP}$-complete version of the Network Loading Problem (NLP). We examine a branch and bound solution approach tailored to the NLP formulation through the use of three families of cutting planes and four heuristic procedures. Theoretically, we specify facet defining conditions for a cut family that we refer to as odd flow inequalities and show that each heuristic yields a 2-approximation. Computationally, the cheapest of the four heuristic values achieved an average error of 1.3% among solved test problems randomly generated from realistic data. The branch and bound method solved to optimality 67% of these problems. Application of the cutting plane families reduced the average relative difference between upper and lower bounds prior to branching from 18.8% to 6.4%.**

Keywords— **network loading, network design, cutting planes**

I. Introduction

MOST tractor trailers consist of a cab and a single trailer about 48 feet long, but some cabs can accommodate in tandem up to two relatively short semitrailers, each about 28 feet long, known as pups. See Figure 1.

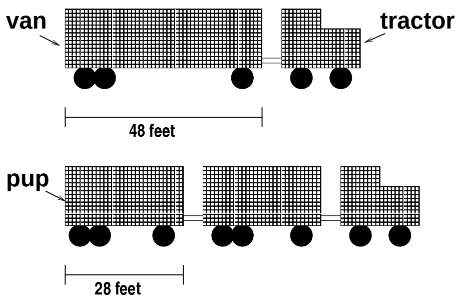

Fig. 1. **A conventional tractor trailer compared with a "tandem" of two semitrailers known as pups.**

In these situations, the cost to a carrier of towing two pups from one location to another is essentially the same as that of towing just one along the same route, half that of towing either three or four, and so forth. Pup matching is the problem of minimizing these stepwise discontinuous costs by matching or pairing pups behind cabs in the most efficient manner.

As an example, in the shipping network represented in Figure 2, the arc lengths represent the cost of sending a cab towing one or two pups from the terminal represented by the tail node to the terminal represented by the head node. Suppose that a carrier must send one pup from node 1 to node 4 and a second pup from node 2 to node 4. If each cab could tow only one pup, it would be optimal to send each pup along its shortest path and incur a cost of 5 for each. However, since pups can be paired, the carrier can achieve the optimal cost of 9 by sending both pups singly to node 3 and then pairing them to the same cab along the arc from node 3 to node 4.

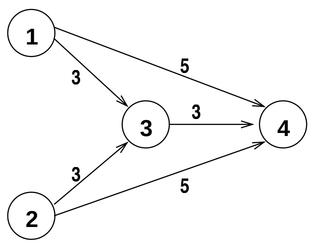

Fig. 2. **Arc lengths represent the cost of sending a cab along with one or two pups from the tail node to the head node.**

Trucking is a large industry. As reported by the Department of Transportation, in 1998 the U.S. trucking industry had revenues of just under $200 billion, and its 7.7 million trucks carried over a trillion ton-miles of freight. Therefore, even modest percentage gains in operational efficiency can translate into substantial monetary savings. Pups not only provide increased capacity over conventional tractor-trailers but also provide flexibility to shift pups among cabs. Consequently, it seems worthwhile to study the problem of optimally deploying this flexibility.

Barnhart and Ratliffe [1] have modelled and efficiently solved two different truck/rail intermodal trailer routing problems. Both problems consider full length trailers. However, the latter resembles pup matching since its rail costs are per flatcar, and each flatcar can accommodate up to two trailers. Each origin-destination path, though, includes at most one arc over rail. Consequently, each trailer travels paired with at most one other trailer and a weighted matching algorithm can solve the problem. The matching problems that we formulate permit each trailer to pair with a different trailer over each arc of its O-D path, and direct application of a matching algorithm cannot solve the problem.

Barnhart and Kim [2] developed an integer programming formulation of a specific pup matching problem they refer to as the core inter-group line-haul problem. This problem involves construction of driver routes to service requested pickups and deliveries of pups at the End-of-Line terminals associated with a single Consolidation Center within a logistics network. They proposed an approximate solution approach that uses two weighted matching subroutines and

J. M. Bossert, M.I.T. Operations Research Center, Cambridge, MA, 02139, USA. email: bossert@mit.edu.
T. L. Magnanti, M.I.T. School of Engineering. email: magnanti@mit.edu.

demonstrated the effectiveness of this approach using both randomly generated data and data provided by a large LTL (less than truckload) carrier. Both their formulation and solution approach permit infeasibilities that we describe as waiting rings.

II. Formulation, Notation, and Complexity

This section outlines assumptions used to reduce the real pup matching problem to a concise problem statement. Section II-A describes the modelling assumptions, and II-B states the resulting problem in instance-problem format. Section II-C presents an initial but incomplete integer programming formulation.

A. Modeling Assumptions

We assume that the motor carrier in question operates on a well defined logistics network that can be expressed adequately as a directed graph with a known cost of sending a driver, as well as one or two pups, along each arc of the network. We assume these costs either include or dominate all other relevant costs, including those incurred switching pups from one cab to another. We also assume that each pup is closed before leaving its origin, not opened until reaching its destination, and that the carrier is concerned only with the costs of transporting the closed pups. That is, the problem addresses no load consolidation issues.

The preceding assumptions restrict the scope of the problem. We also make several simplifying assumptions. First, we ignore any time constraints imposed upon the shipment of the pups, and search for the minimum cost shipping strategy that sends the pups to the required destinations. Additionally, we ignore limits on driver and cab resources and assume immediate availability of a loaded cab at the arc tail node. Consequently, we assume that the carrier can move a pup along any outgoing arc of its current node for no cost other than that attributed to moving along the arc (the marginal cost of which might be 0). The adequacy of these latter assumptions depends on the application.

Within this framework, we might consider two problem variations. The first requires shipment of a pup between a specified origin-destination pair. The second variation requires that each destination node receive one or more pups, but without regard to their origin, perhaps because each pup contains the same commodity. The second variation identifies but does not pair origin and destination nodes. We consider the former variation the primary case, and consider it exclusively in the remainder of this paper.

B. Problem Statement

The preceding assumptions lead to the following problem statement.

Pup Matching
Instance: A directed network $G = (N, A)$, a set K of pairs of elements of N, and a cost function $c : A \rightarrow \mathcal{R}+$.
Problem: Find the minimum cost loading of capacitated facilities and an assignment of a unit flow to a path from the first to the second node of each of the pairs K. Each unit of loading on arc $a \in A$ costs $c(a)$ and permits 1 unit of flow or 2 units flowing together to traverse arc a.

The "togetherness" requirement in the problem statement reflects the fact that two units of flow must be available on an arc simultaneously to be able to use a single unit of loaded capacity. (See our later discussion of waiting rings.) A feasible loading permits specification of an origin-destination path for each pup, and for each arc of such a path, an indication of another pup, if any, that travels with it behind the same cab. We term such a specification a *routing.* Note that a routing includes both paths and pairs. Feasibility of a loading and an accompanying routing corresponds to the existence of a dispatching sequence of the loaded cabs that implements the pup routing. We refer to two pups assigned to traverse one or more arcs together as *pairs* or *matches.* We use the latter two terms interchangeably.

This problem statement permits a pup to be matched to more than one other pup and over more than one arc. As a result, matching costs are not well defined, and we cannot solve this problem by directly applying a weighted nonbipartite matching algorithm. In fact, Pup Matching is at least as hard as Three Dimensional Matching and so $\mathcal{NP}$-complete.

Theorem 1: Pup Matching, posed as the decision problem of determining whether a feasible cab loading with cost no greater than a specified value exists, is $\mathcal{NP}$-complete.

C. Integer Programming Formulation

We formulate Pup Matching as a special case of the Network Loading Problem (see Magnanti, Mirchandani, and Vachani [3], [4]), with pups as commodities and each loading, in the form of a driver or cab, providing 2 units of capacity. The model includes the following data:
$G = (N, A)$: the shipping network,
c_{ij} : cost to send one cab, as well as one or two pups, on arc $(i, j) \in A$,
O_k, D_k : origin and destination nodes, respectively, for pup $k, k = 1, 2, \ldots K$,

and the following variables:
f_{ij}^k : binary variable, with a value of 1 indicating that pup k is routed on arc (i, j),
z_{ij} : integer variable, the number of cabs assigned to arc (i, j).

Using this notation, we can formulate the model as follows.

NLP formulation of Pup Matching
minimize:

$$\text{cost} \quad = \quad \sum_{i,j \in A} c_{ij} z_{ij} \tag{1}$$

subject to:

$$\sum_{j \in N} f_{ij}^k - \sum_{j \in N} f_{ji}^k \quad = \quad \begin{cases} 1, & \text{if } i = O_k \\ -1, & \text{if } i = D_k \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\sum_{k\le K} f_{ij}^k \quad \le \quad 2z_{ij} \qquad (3)$$

$$z_{ij} \quad \ge \quad 0, \text{ integer} \qquad (4)$$

$$f_{ij}^k \quad \text{binary} \qquad (5)$$

The objective (1) minimizes cab loading cost. Constraints (2) enforce pup flow balance for each pup at each node, and constraints (3) require sufficient arc capacity. Constraints (4) and (5) enforce nonnegative and binary integrality, respectively.

The NLP formulation fails to explicitly enforce the constraint that both capacity units of a cab loading not be used separately, since it permits two pups traversing an arc separately to each exhaust one unit of capacity. That is, the formulation implicitly assumes that two pups assigned to the same arc can always be matched to a single cab. Example 1 illustrates that this assumption is not necessarily valid, and that, as a consequence, it might not be possible to implement a Pup Matching solution for the optimal cost of its NLP formulation.

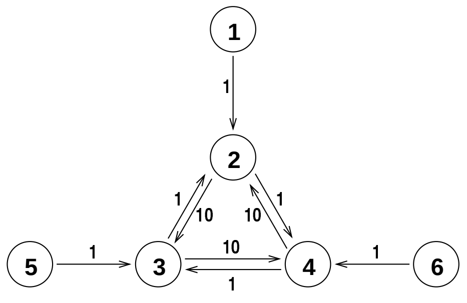

Fig. 3. Shipping network for which the NLP Pup Matching formulation fails. Node numbers and arc costs are specified.

Example 1: 3 pups are to travel on a network with topology and arc costs as shown in Figure 3. Pup A is to travel from node 1 to node 3, pup B from node 6 to node 2, and pup C from node 5 to node 4.

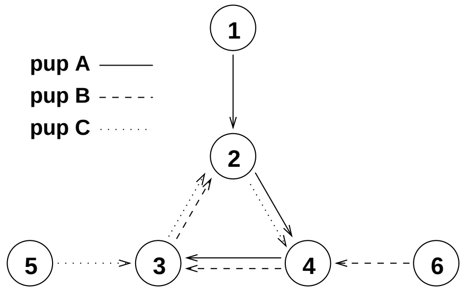

Fig. 4. Optimal routing to the NLP formulation of Example 1.

Figure 4 depicts an optimal routing determined by the flow variables of the NLP Pup Matching formulation that requires only 1 cab between each pair of nodes joined by an arc. Note that when pup A reaches node 2, it must wait for pup C if the routing is to be implemented for the loaded capacity. Similarly, when pup C arrives at node 3, it must wait for B. Finally, when pup B arrives at node 4, it must wait for A. Breaking this gridlock requires allocation of additional cabs.

Definition 1: Suppose that a pup A has arrived at some node but cannot advance along its assigned path until its assigned pair, B, for the next arc of that path has also arrived. Suppose further that B must wait at its present node until some other pup, C, has arrived, and similarly, pup C must wait for pup D, pup D for pup E ... pup Q for pup R. If this precedence chain closes in the sense that pup R waits upon pup A, none of the pups in the chain can advance according to the assigned routing, and the routing is thus infeasible. We refer to the pups involved in this gridlock and the portion of each such pup's origin-destination path between the node where it waits and the node where it completes travel with the pup that waits on it, as a *waiting ring.* The waiting ring of Figure 4 is defined by pups A, B, and C, and their subpaths among nodes 2, 3, and 4.

Note that a waiting ring is a property of a routing and is independent of the dispatch sequence and travel times. In Example 1, no matter how quickly pup A arrives at node 2 relative to pups B and C, it cannot advance according to the assigned routing until pup C arrives at node 2, and pup C never arrives at node 2.

We show that waiting rings account fully for the incompleteness of the NLP formulation of Pup Matching, and, furthermore, that if all pups share a common origin or common destination, waiting rings can be removed from an NLP solution without incurring additional cost.

Corollary 1: The NLP formulation of Pup Matching determines the optimal loading cost if all pups share a single origin or destination.

Our proof of Theorem 1 implies that Pup Matching remains $\mathcal{NP}$-complete in the single origin case. Consequently, the NLP formulation itself is $\mathcal{NP}$-complete.

Corollary 2: The NLP formulation of Pup Matching, posed as the decision problem of determining whether a feasible solution with cost not exceeding a specified value exists is $\mathcal{NP}$-complete.

Finally, we show $\mathcal{NP}$-completeness of the decision problem of whether a ring free routing corresponds to a given feasible solution to an NLP formulation of Pup Matching. We refer to the problem as the Waiting Ring Problem and state it as follows.

Waiting Ring Problem

Instance: A directed network $G = (N, A)$, a set of k (acyclic) paths on G, and an integral capacity loading on each arc in A such that the number of paths traversing each arc is no more twice the loading on that arc.

Problem: If each unit of loading can be used once to advance one or two tokens along its assigned arc, determine whether there exists a utilization sequence of the loadings that advances one token from the head node to the tail node of each of the k paths.

Theorem 2: The Waiting Ring Problem is $\mathcal{NP}$-complete.

This $\mathcal{NP}$-completeness implies (see Karp and Papadimitriou [5]) that we cannot reasonably expect to determine a set of linear inequalities that eliminates waiting rings from the NLP formulation. The following solution approach and computational study consider only the NLP formulation of Pup Matching. The implications of Theorem 2 and our observation of few waiting rings on initial Pup Matching test instances seem to justify our focus on this incomplete formulation.

III. Branch and Bound Solution Approach

This section summarizes our adaptation of the branch and bound solution approach to the Network Loading formulation of Pup Matching. Section III-A motivates the need for specialization by describing our initial computational difficulties. Section III-B describes heuristic solution approaches and approximation results, and Section III-C presents valid inequalities that we use as cuts.

A. City Blocks Test Problem

To assess the difficulty of the NLP formulation of Pup Matching, we first applied the default CPLEX branch and bound routine to a series of fabricated problems including several defined on the grid-like graph shown in Figure 5. The graph represents a set of city blocks, and each edge in the figure corresponds to two arcs, one in each direction.

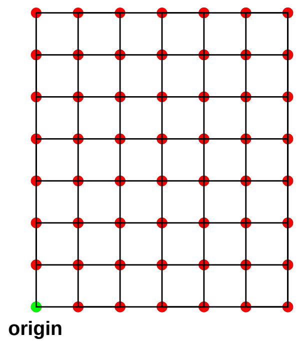

Fig. 5. The underlying graph for several Pup Matching test problems.

Example 2: Deliver a pup from the origin node indicated in the lower left corner of Figure 5 to each of the other 55 nodes. Each arc cost is 1.

The objective equals the number of cab loadings needed to complete the deliveries. Given this problem, we might quickly find a solution of cost 196 similar that shown in Figure 6: the horizontal flow occurs only on the lower most lateral street, and the numbers indicate cab loadings. Although 196 is the optimal solution, the unmodified branch and bound code was able to improve its lower bound from the LP relaxation value of 182 to only 184 with several days of computation time.

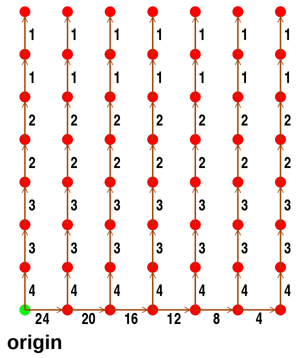

Fig. 6. Solution of cost 196 to the problem of delivering 1 pup from the origin node to each of the other 55 nodes. The numbers indicate cab loadings.

B. Heuristics and Approximation Algorithms

To find initial solutions and, hopefully, high quality upper bounds, we developed four heuristic procedures. The first considers Pup Matching assuming that a pup can be paired with at most one other pup. Given that pups A and B are paired once we have imposed this additional single pairing restriction, we can determine their optimal routing, independent of the routing of other pups, by solving the pup matching problem defined on the same network and involving only those two pups. We can find the optimal routing for two pups, and so a matching cost for the two pups, using a series of Steiner calculations in $\mathcal{O}(|\ N\ |^3)$ time. Consequently, we can find the optimal solution to Pup Matching assuming the single pairing constraint using a weighted matching algorithm. We refer to the procedure of solving each possible 2 pup problem and then applying a matching algorithm to the resulting cost matrix as the Matching Approximation.

Theorem 3: Pup Matching under the additional constraint that each pup may be paired with at most one other pup can be solved in polynomial time.

The Matching Approximation solution value can be no greater than twice the optimal Pup Matching solution, since the heuristic can do no worse than routing each pup singly on its shortest origin-destination path. On the other hand, examples show that the heuristic solution value can be arbitrarily close to twice the optimal value.

Theorem 4: The absolute performance ratio of the Matching Approximation is 2.

We also consider three heuristics based on shortest path calculations. The simplest routes each pup on its shortest path. The second iteratively routes a pup on its shortest path and then updates arc costs to reflect marginal costs. The third method combines the other two by routing the first several pups on their shortest paths using original costs and the remaining using marginal costs. The cost shift might prevent squandering 0 marginal cost arcs on pups routed early in the procedure.

Clearly, none of these procedures is optimal. Furthermore, the three heuristics do not necessarily output feasible Pup Matching solutions since each might produce a waiting ring. However, each generates a solution feasible to the NLP formulation and provides a 2-approximation to that problem.

Theorem 5: Each of the three successive shortest path heuristics provides a 2-approximation for the NLP formulation of Pup Matching.

C. Valid Inequalities

To tighten the lower bound provided by the LP relaxation of the NLP formulation, we append cuts from three families of valid inequalities — cutset inequalities, residual capacity inequalities, and a new class that we refer to as odd flow inequalities.

Cutset inequalities (see Magnanti, Mirchandani, and Vachani [4] and Barahona [6]) ensure that the capacity loaded across a cut is sufficient to accommodate the flow that must cross the cut. For Pup Matching, they assume the following relatively simple form:

$$\sum_{i\in S, j\notin S} y_{ij} \geq \lceil \frac{D_S}{2} \rceil, \forall S \subset N. \qquad (6)$$

In this expression, D_S is the number of pups that must leave node set S, that is, the number of pups with origin in S and destination in $N\backslash S$. The left side of the inequality is the number of cabs loaded on the cut defined by nodes S. This quantity is integral and each cab has capacity 2. Consequently, the loading must be at least the ceiling of half the net demand. Since we are unable to efficiently solve the cutset separation problem, we append the inequality for each cut defined by a single node and then, as in Balakrishnan, Magnanti, Sokol, and Wang [7], iteratively use a Gomory-Hu tree to identify other promising cuts.

A residual capacity inequality (see Magnanti, Mirchandani, and Vachani [3], [4]) constrains the loading requirement on a single arc. One exists for every arc-commodity subset combination. For the Network Loading formulation of Pup Matching, the residual capacity inequalities reduce to:

$z_{ij} \geq \sum_{k\in L} f_{ij}^k - \lfloor \frac{|L|}{2} \rfloor$,

for an odd cardinality subset of pups L. The residual capacity constraint for an even cardinality subset reduces to the arc capacity inequality (3).

Atamturk and Rajan [8] have shown how to separate the residual capacity inequalities for a single arc of a Network Loading Problem with k commodities and with facilities of an arbitrary capacity in $\mathcal{O}(k)$ time. We can separate the residual capacity inequalities for Pup Matching in $k\log k$ time by directly checking the inequality for commodity subsets L of maximum flow for each possible odd cardinality, since the RHS is maximized by the commodity subset defined by the largest f_{ij}^k values.

Lemma 1: A given fractional solution violates the residual capacity inequality for a given arc of a Pup Matching problem only if it violates the inequality for a commodity subset L of maximum flow for some odd cardinality $\mid L \mid$.

Although the cutset and residual capacity inequalities improve the lower bounds, they do not lead to efficient solutions of all the city blocks test problems, including Example 2. In trying to prove by other means optimality of the Example 2 solution of value 196 diagrammed in Figure 6, we discovered a set of inequalities that constrain flow on arcs incident to a node with odd demand.

If total pup flow on an arc is odd, some capacity loaded on that arc must remain unused and we could tighten its capacity constraint. In general, the flow on a given arc could be even or odd. However, if the net demand at a node is odd, then the total inflow or total outflow must be odd, and the node must be incident to at least 1 unit of unused pup capacity, $\frac{1}{2}$ a cab's worth. odd flow inequalities exploit this observation to tighten the sum of arc capacity constraints over the set of arcs incident to a node of odd demand.

Theorem 6: The following odd flow inequalities are valid for the NLP formulation of Pup Matching for each node $i \in N$ with odd net demand:

$$\sum_{a\in A_i} z_a - \frac{1}{2}\sum_{k\in K}\sum_{a\in A_i} f_a^k \geq \frac{1}{2}. \qquad (7)$$

In this expression, A_i denotes the set of arcs incident to node i.

Under strong connectivity conditions, the odd flow inequalities define facets of the convex hull of the Pup Matching polyhedron.

Theorem 7: If $G = (N, A)$ is strongly connected (contains a directed path between each pair of nodes), the total net demand of some node $i \in N$, is odd, and node i and those nodes adjacent to it form a clique, then the corresponding odd flow inequality defines a facet of the Pup Matching polyhedron.

Though our solution procedure appends odd flow inequalities corresponding to only single nodes, the same logic applies to any subset of nodes with odd net demand.

IV. Computational Results

Figure 7 summarizes the results of our solution procedure on five city blocks problems. Problem 7i is the city blocks problem of Example 2. 7ii and 7iii are defined on the same graph. Problems 9i and 9ii are defined on a similarly sized graph of one way streets. The portion of the graph below the zero line depicts the error of the best heuristic. In all cases except 9ii, at least one heuristic found the optimal solution, and, in that case, the best value was less than 2% from optimal. The portion of the graph above the zero line summarizes the lower bound improvement from sequential application of the cutting plane families. The length of each composite box is proportional to the LP relaxation error, and each inner box indicates the bound improvement from the corresponding family of inequalities. In Problem

9ii for example, the LP relaxation error was 12.8%, the cutset inequalities reduced the error to 8.0%, the residual capacity inequalities reduced the error about another 1%, and the odd flow inequalities increased the lower bound to the optimal solution value.

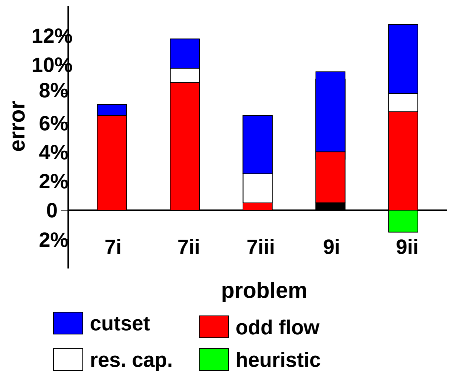

Fig. 7. Results of the branch and bound procedure on city blocks problems. The portion of the graph above the zero line depicts lower bound improvement from sequential application of the three cutting plane families, and the portion below the zero line indicates the error of the best heuristic.

We also applied the branch and bound solution procedure to 30 problems randomly generated from realistic data. The results seem good but not as dramatic as those exhibited for the city blocks problems. Given a node set in (latitude, longitude) format based on a real logistics network, we defined problems by selecting a subset of nodes, calculating arc lengths as Euclidean distances, and randomly selecting origin-destination pairs. Since we used the distance metric, all the underlying graphs were complete. About half the problems had a single origin.

We limited the branch and bound tree to 220M of memory and 2 hours of CPU time. Using all three cut families, we were able to solve 67% of the problems to optimality with an average gap reduction of 18.8% to 6.4%. (Since the procedure did not solve all the problems, the gap reflects the difference between a lower bound and the tightest upper bound.) Without the odd flow cuts, we were able to solve 30% of the problems and reduced the gap to 7.8% on average. With no cuts, we solved only 17% of the problems. Among the solved problems, the average heuristic error was 1.3%.

V. Conclusions

We have investigated four heuristic methods and a cutting plane based branch and bound procedure for solving the pup matching problem. Among the more realistic test problems that we could solve to optimality, the heuristics performed very well, obtaining solutions with objective values within 1.3% of optimal. To what extent we are witnessing a selection bias (that is, whether the heuristics were more effective for problems we have been able to solve) remains to be seen.

Even though the heuristic methods are able to generate good feasible solutions, because of weak linear programming lower bounds, a default implementation of branch and bound was not able solve problems to optimality within reasonable running times. Consequently, as in other application settings, our computational study underscores the importance of high quality lower bounds to provably solve integer programs. To this end, the odd flow inequalities have proven very effective. Their discovery permitted us to solve in seconds city blocks problems that we were previously unable to solve with days of computation time. The concept of odd flow inequalities generalizes for a single facility Network Loading Problem with arbitrary facility capacities C, instead of 2, to exploit the observation that the loading on any arc whose total flow is not a multiple of C requires spare capacity. We suspect that cuts based on similar parity arguments would help solve other network design problems.

Acknowledgment. We are grateful to the Singapore-MIT Alliance for providing partial financial support for this research.

References

[1] Cynthia Barnhart and H. Donald Ratliff, “Modeling intermodal routing,” Tech. Rep. COC-91-11, Georgia Institute of Technology, 1991.

[2] Cynthia Barnhart and Daeki Kim, “Routing models and solution procedures for regional less-than-truckload operations,” *Annals of Operations Research*, vol. 61, pp. 67–90, 1995.

[3] Thomas L. Magnanti, Prakash Mirchandani, and Rita Vachani, “The convex hull of two core capacitated network design problems,” *Mathematical programming*, vol. 60, no. 2, pp. 233–250, 1993.

[4] Thomas L. Magnanti, Prakash Mirchandani, and Rita Vachani, “Modeling and solving the two-facility capacitated network loading problem,” *Operations research*, vol. 43, no. 1, pp. 142–157, 1995.

[5] Richard M. Karp and Christos H. Papadimitriou, “On linear characterizations of combinatorial optimization problems,” in *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, 1980, pp. 1–9.

[6] Francisco Barahona, “Network design using cut inequalities,” *SIAM Journal on Optimization*, vol. 6, no. 3, pp. 823–837, 1996.

[7] Anantaram Balakrishnan, Thomas L. Magnanti, Joel S. Sokol, and Yi Wang, “Spare capacity assignment for line restoration using a single facility type,” To appear in *Operations Research*, preprint, 2000.

[8] Alper Atamturk and Deepak Rajan, “On splittable and unsplittable flow capacitated network design arc-set polyhedra,” Unpublished, July 2000.