OPERATIONS RESEARCH CENTER

Working Paper

Pup Matching: Model Formulations and Solution Approaches

by

John M. Bossert and
Thomas L. Magnanti

OR 366-03 **February 2003**

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Pup Matching: Model Formulations and Solution Approaches

John M. Bossert and Thomas L. Magnanti

February 2003

We model Pup Matching, the logistics problem of matching or pairing semitrailers known as pups to cabs that are 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. Branch and bound 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%.

1 Introduction

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.

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 known as pups, each about 28 feet long. See Figure 1. 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.

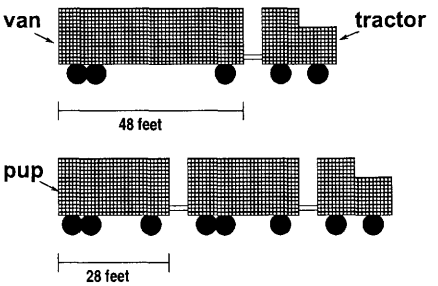

Figure 1: A conventional tractor trailer and a "tandem" of two semitrailers known as pups.

As an example, in the shipping network of 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 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 for a cost of 5 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.

In more general situations, a cab might be paired with different pups, dropping and adding pups at nodes along its route. Pups provide not only increased towing capacity over conventional tractor-trailers, but also greater flexibility through options to shift pups among cabs. The problem of optimally deploying this flexibility seems

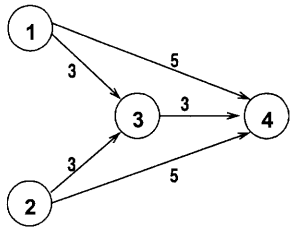

Figure 2: Arc lengths represent the cost of sending a cab towing one or two pups from the tail node to the head node.

worth studying.

Barnhart and Ratliff [6] modeled 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 problems that we examine 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. However, the Matching Approximation that we introduce in Section 3 is similar to the solution technique of Barnhart and Ratliff.

Barnhart and Kim [5] developed an integer programming formulation of a specialized pup matching problem they refer to as the core inter-group line-haul problem. This problem involves construction of cyclic driver routes to service trailer pickups and deliveries at the end-of-line terminals associated with a single consolidation center within a logistics network. That is, drivers or, equivalently, cabs must be routed over circuits within the network so that the corresponding towing capacity permits each

pickup trailer to advance from its origin to the consolidation node and each delivery trailer to advance from the consolidation node to its destination node. The objective is linear in the number of cabs traversing each arc. Barnhart and Kim proposed an approximate solution approach that uses two weighted matching subroutines, and they 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* in Section 2.

Li, McCormick, and Simchi-Levi [12] considered a class of problems more general than our pup matching model that they refer to as point-to-point delivery and connection problems. The problems involve sending a single item from each of p origins to p destinations. Up to C items at once can share each unit (typically a truck) of capacity installed or loaded on an arc, and costs are linear in the number of such capacity units loaded. Pup matching under our assumptions corresponds to the special case of $C = 2$. The authors considered problems with prepaired or fixed and unfixed origins and destinations, on both directed and undirected graphs. They also considered the special cases with large values of capacity C, the problems of connecting origins and destinations as cheaply as possible. The authors showed that all variations are strongly $\mathcal{NP}$-hard, and they described a polynomial time algorithm for the special case of point-to-point delivery with a fixed value of p.

The rest of the paper is organized as follows. Section 2 describes modeling assumptions, incompletely formulates the resulting problem as a special case of the network design problem known as Network Loading, and attributes the incompleteness to

waiting ring infeasibilities. Section 3 describes the heuristics and valid inequalities employed by our branch and bound solution approach, and Section 4 summarizes computational results. Finally, Section 5 offers conclusions and poses some research questions.

2 Formulation, Notation, and Complexity

2.1 Modeling Assumptions

We assume that the motor carrier in question operates on a well defined logistics network that is adequately modeled as a directed graph with a known cost of sending a cab and driver, as well as one or two pups, along each arc of the network. We assume these costs 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 with only the costs of transporting the closed pups. That is, the problem addresses no load consolidation issues.

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 such as driver availability and cab rebalancing. We effectively assume immediate availability of a loaded cab at each arc tail node, and, in turn, that the carrier can move a pup along any outgoing arc of its cur-

rent node for no cost other than that attributed to moving along the arc, the marginal cost of which might be 0. The model of Barnhart and Kim [5] requires cyclic routing of each cab and enforces net trailer balance at each node. A model might require such constraints to satisfy driver work rules or to ensure a longer term deployment of resources capable of meeting future shipping requirements. The adequacy of our simplifications depends on the application, but the model hopefully captures at least a core structure common to this family of problems.

Within this modeling framework, we might consider two problem variations. The first requires shipment of a pup between a specified origin-destination pair. The second requires that each destination node receive one or more pups, but without regard to their origin, perhaps because each trailer contains the same commodity. Like a standard network flow problem, this 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.

2.2 Problem Statement

The problem statement refers to the collective towing capacity allocated over a network as a "loading." A feasible loading of capacity permits specification of an origin-destination path for each pup, and for each arc of this path, an indication of another pup, if any, that travels with it behind the same cab. We term such a specification a *routing*. 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.

The preceding assumptions lead to the following problem statement.

> **Pup Matching (PM)**
> **Instance:** A directed network $G = (N, A)$, a set of K ordered pairs of elements of N, and a cost function $c : A \rightarrow \mathcal{R}+$.
> **Problem:** Find the minimum cost capacity loading of G that permits a multicommodity flow with one unit flow from the first to the second node of each of the K pairs. Each unit of capacity loaded 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 reflects the fact that two trailers must be available at a tail node simultaneously to share a single unit of loaded capacity. Our discussion of waiting rings in Section 2 details the difficulty of accounting for these constraints.

A pup may have more than one pair along its origin-destination path. As a result, pairwise 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 whether some feasible cab loading costs no more than a specified value, is $\mathcal{NP}$-complete.*

Proof: See [12] or [8].

Li, McCormick, and Simchi-Levi [12] proved the same complexity result by a transformation of 3-Satisfiability. The proof in [8] establishes additional results, for example, the problem remains $\mathcal{NP}$-complete for situations with a single origin or destination.

2.3 Integer Programming Formulation and Waiting Rings

We formulate Pup Matching as a special case of the Network Loading Problem (see, for example, Magnanti, Mirchandani, and Vachani [13], [14], Barahona [4], or Bienstock and Günlük [7]) that casts pups in the role of commodities and cabs in the role of capacity providing facilities. The model includes the following data:

$G = (N, A)$: the (directed) 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, $k \in \{1, 2, \ldots K\}$,

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 loaded on arc (i, j).

Using this notation, we formulate the model as follows.

NLP formulation of Pup Matching
minimize:

$$\sum_{\{i,j\} \in A} c_{ij} z_{ij} \tag{1}$$

subject to:

$$\sum_{j \in N} f_{ij}^k - \sum_{j \in N} f_{ji}^k = \left\{ \begin{array}{cc} 1, & \text{if } i = O^k \\ -1, & \text{if } i = D^k \\ 0, & \text{otherwise} \end{array} \right\}, \forall i \in N, k \in \{1, 2, \ldots K\}, \tag{2}$$

$$\sum_{k \leq K} f_{ij}^k \leq 2 z_{ij}, \forall (i, j) \in A \tag{3}$$

$$z_{ij} \geq 0, \text{ integer}, \forall (i, j) \in A \tag{4}$$

$$f_{ij}^k \quad \text{binary}, \forall (i, j) \in A, k \in \{1, 2, \ldots K\}. \tag{5}$$

The objective (1) minimizes the 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.

The flow variables f define pup paths but not pairings, so a solution to the NLP formulation typically corresponds to many routings. Also, the NLP formulation fails to explicitly enforce the constraint that a cab loading must use both units of its capacity together, since it permits two pups traversing an arc separately to each exhaust one unit of capacity and so effectively share a cab. That is, the formulation implicitly assumes we can match to a single cab two pups assigned to the same arc. Example 1 illustrates that this assumption is not necessarily valid, and that, as a consequence, a feasible solution to the NLP formulation might not allocate enough capacity to implement a Pup Matching solution.

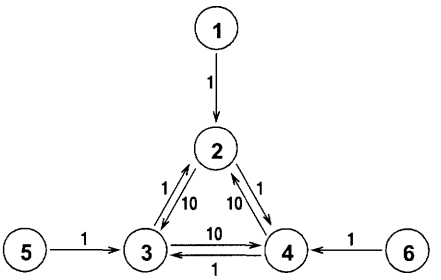

Figure 3: The optimal solution to the NLP Pup Matching formulation can be infeasible to the Pup Matching Problem. Arc numbers are cab travel costs.

Example 1 *Consider a 3 pup example on a network with topology and arc costs 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.*

Figure 4 depicts an optimal routing determined by the flow variables of the NLP Pup Matching formulation that requires only 1 cab on each arc crossed by a pup path. 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. After reaching

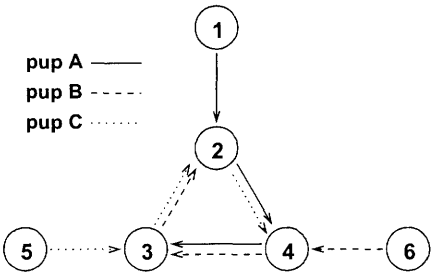

Figure 4: Optimal routing to the NLP formulation of Example 1. The solution assigns a single cab to each arc crossed by a pup path, yet each pup can advance only one arc.

the inner triangle in the figure, no pup can advance while matched with another pup. Breaking this gridlock requires allocation of additional cabs.

The following definition generalizes this class of infeasibilities.

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.*

A waiting ring is a property of a routing and is independent of travel times and the dispatch sequence used to implement the routing. 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. Also, we can assume the pups forming a ring are distinct by closing a ring upon first hitting a particular pup a second time. The *time constraints* of Li, McCormick, and Simchi-Levi eliminate waiting rings. However, their paper did not formulate

these constraints mathematically, as it did not formulate the point-to-point delivery problems as network loading problems.

As we show next, waiting rings account fully for the discrepancies between the NLP formulation of Pup Matching and the combinatorial problem we stated at the beginning of this section, in the sense that if we can construct a ring free routing (paths and pairings) from an NLP solution, we can also construct a cab dispatching sequence that demonstrates feasibility to the combinatorial problem.

Theorem 2 *If some routing of a solution to the NLP formulation of Pup Matching contains no waiting ring, the solution is feasible to Pup Matching itself.*

Proof: Assume a ring free routing and imagine dispatching cabs along arcs to advance each pup. Say that a pup can advance from its current node i if it is to cross the next arc of its path singly or if its pair for that next arc has also arrived at node i. We can find a pup that can advance by arbitrarily selecting a pup that has not yet reached its destination, checking whether it can advance, if not, checking whether the pair it waits for can advance, and so on, along the chain of pairing relationships. Since the assumed routing contains no ring, no such search will cycle among the pups, and each search will identify a pup that can advance. Each pup will eventually reach its destination since its path is finite. □

This dispatching result does not indicate how to construct a routing. A routing specifies a path for each pup, while the flow variables for an NLP solution might trace a cyclic walk. We can remove any such cycles without introducing infeasibility, since we can convert any dispatching sequence corresponding to the solution before removing any cycle to a routing of the acyclic solution by sending singly any former pairs of a pup within a cycle. The latter routing does not require additional cab loadings, and, with no new pup pairings, cannot create a new waiting ring. Given an acyclic NLP solution, we can construct a routing by assigning matchings for each arc with a flow of more than one pup. However, Theorem 4 below shows that the

problem of determining whether some ring free routing corresponds to such an acyclic NLP solution is $\mathcal{NP}$-complete.

If all pups share a common origin or destination, we can modify a routing to remove rings without an increase in cost by essentially relabeling the pups at the ring.

Theorem 3 *If all pups share a common origin (destination), we can eliminate waiting rings without an increase in cost.*

Proof. Suppose a solution contains a waiting ring. Each pup involved in the ring can advance by maintaining its current ring to destination (origin to ring) subpath and taking the origin to ring (ring to destination) subpath previously assigned to the pup in the ring that waits on it (that it waits on). This reassignment breaks the ring since each involved pup can complete its subpath within the ring singly with no additional (and possibly fewer) cabs. Furthermore, this modification does not alter the cab requirements outside the ring. □

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

As previously noted, the proof of Theorem 1 in [8] implies that Pup Matching remains $\mathcal{NP}$-complete in the single origin case. Since the NLP formulation determines the optimal solution in this special case, it is also $\mathcal{NP}$-complete.

Corollary 2 *The NLP formulation of Pup Matching, posed as the decision problem of whether some feasible solution has a cost not exceeding a specified value, is $\mathcal{NP}$-complete, even in situations with a single origin or destination.*

Since we can usually construct many routings from an NLP solution, a single ring does not imply infeasibility of the underlying NLP solution. In fact, the decision problem of whether we can construct a ring free routing from a given solution feasible to the NLP formulation is $\mathcal{NP}$-complete. We refer to this problem as the following Waiting Ring Problem.

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 satisfying the condition 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 some utilization sequence of the loadings advances one token from the head node to the tail node of each of the K paths.

Theorem 4 *The Waiting Ring Problem is $\mathcal{NP}$-complete.*

Proof: See [8]. □

Suppose we could search in nondeterministic polynomial time some set of inequalities that eliminates waiting rings from the NLP formulation. We could then solve the complement to the Waiting Ring Problem (i.e., the problem of whether every routing contains a ring) by checking all such inequalities. Consequently, these inequalities would imply that the Waiting Ring Problem is in co-$\mathcal{NP}$ as well as $\mathcal{NP}$, and so the inequalities most likely do not exist. (See Karp and Papadimitriou [11].)

The following solution approach and computational study consider only the NLP formulation of Pup Matching. The complexity implications of Theorem 4 and our observation of few waiting rings on initial Pup Matching test instances seem to justify our focus on this incomplete formulation. Moreover, as a relaxation of Pup Matching that permits waiting rings, the NLP solution always provides a lower bound on the optimal Pup Matching cost.

3 Branch and Bound Solution Approach

3.1 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 that might represent a set of city blocks. Each edge in the figure corresponds to two arcs, one in each direction.

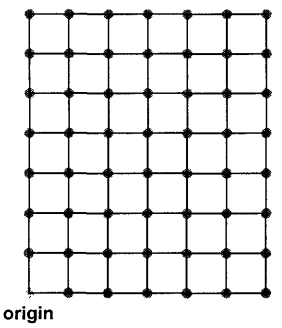

Figure 5: The underlying graph for several Pup Matching test problems.

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 deliver all pups. Given this problem, we might quickly find a solution of cost 196 similar that shown in Figure 6, with the horizontal flow occurring 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 improved its lower bound from the LP relaxation value of 182 to only 184 with several days' computation time.

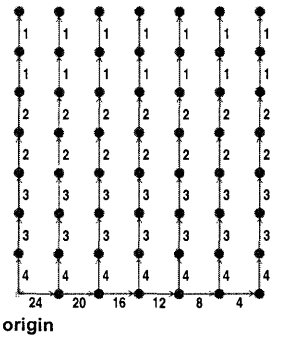

Figure 6: Solution of cost 196 to the problem of delivering 1 pup from the origin node to each of the other 55 nodes. The numbers indicate cab loadings.

3.2 Heuristics and Approximation Algorithms

Pup Matching poses a trade-off between directly routing pups and efficiently utilizing loaded capacity. To find initial solutions and, hopefully, high quality upper bounds, we developed four heuristic approaches to address this trade-off. The first heuristic employs an exact cubic algorithm for matching only two pups to derive pairwise matching costs for finding an optimal Pup Matching solution subject to the additional constraint that we can pair each pup with at most one other pup. The other three heuristics are based upon shortest path calculations. These procedures dynamically modify the arc lengths to steer pups toward unused capacity and so encourage efficient capacity utilization.

3.2.1 Matching Two Pups and the Matching Approximation

We previously noted that a weighted matching algorithm does not necessarily solve Pup Matching because an optimal solution might match a pup to more than one other pup. Matching costs would be well defined and independent, however, if each pup

could be paired with at most one other pup along its entire origin-destination path. In this setting, we would be able to solve Pup Matching by applying a weighted matching algorithm to a graph with a node corresponding to each pup and edge weights given by the optimal matching costs for these two pups. We next describe how to solve the underlying two pup matching problem and then formalize the matching approach for Pup Matching.

Since each cab can tow two pups, matching two pups reduces to a connectivity problem, that is, its optimal solution is the cheapest subgraph with a directed path connecting each origin-destination pair. We next observe that in some optimal solution these two paths merge along at most one subpath.

Lemma 1 *In some optimal solution to the two Pup Matching Problem, the two O-D paths coincide only along some (possibly empty) directed subpath. That is, some optimal solution has nonoverlapping $O_1 - D_1, O_2 - D_2$ paths or the general structure shown in Figure 7, with the arrows representing disjoint paths.*

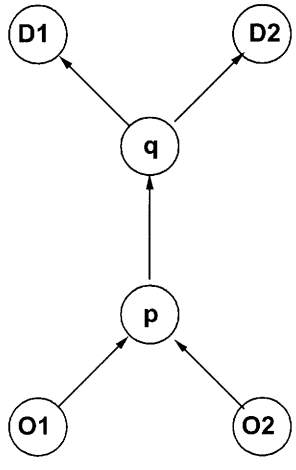

Figure 7: General structure of an optimal solution to two Pup Matching. Each arrow represents a path.

Proof. The result follows from the observation that if the two $O - D$ pup paths contain two distinct directed paths P and Q, between two nodes p and q of the

network, then routing both pups on the cheaper of P and Q costs no more than routing one pup on each path. □

As a result of Lemma 1, the following algorithm yields an optimal solution to the two Pup Matching Problem.

Algorithm 1.

1. Run an all pairs shortest path algorithm on the network. Let $d(i,j)$ be the shortest distance between nodes i and j.
2. Let O_1, O_2, D_1, D_2 be the origin and destination nodes. For each pair of nodes p and q, calculate $l_{p,q} = d(O_1,p) + d(O_2,p) + d(p,q) + d(q,D_1) + d(q,D_2)$.
3. The optimal solution corresponds to $\min\{\min_{p,q}\{l_{pq}\}, d(O_1,D_1) + d(O_2,D_2)\}$.

Proof of Correctness. With given initial and final junction nodes p and q, each optimal subpath $O_1 - p, O_2 - p, p - q, q - D_1$, and $q - D_2$ is, by contradiction, a corresponding shortest path. So, $l_{p,q}$ is the optimal solution value given that the pups travel together only from node p to node q. Now, either the pups travel together or they do not, and, if they do not, it is optimal to send each on its shortest path. Step (3) considers both possibilities. □

The two Pup Matching Problem is similar to the Directed Steiner Network Problem, that of finding a minimum cost subgraph containing a directed path from a specified source node to each node in a specified subset of nodes (see, for example, Winter [15]). If the two pups share a common origin or destination, Pup Matching reduces to the Steiner Problem. The general two pup problem is a special case of the generalized Steiner Network Problem of finding the cheapest forest that connects each of a specified set of node pairs.

The following algorithm formalizes the approach of applying a weighted matching routine to the costs determined by solving two Pup Matching Problems. The algorithm solves Pup Matching under the additional constraint that we may pair each pup with at most one other pup.

Matching Approximation (MA).

1. For every pair of pups, solve the two pup problem with the Algorithm 1.
2. Using the results of step 1, form a cost matrix $C = (c_{pq})$. If the number of pups is odd, add a dummy pup with matching costs equal to the shortest origin-destination path length of the matched pup.
3. Solve the weighted nonbipartite perfect matching problem defined by the cost matrix C calculated in step 2.

Corollary 3 *Pup Matching under the additional constraint that each pup may be paired with at most one other pup is polynomially solvable.*

A pup matched with the dummy is paired with no real pup and travels along its shortest path. Each other pup is routed according to the solution of a two pup problem. Some such pups might be matched in name only and actually travel singly, as specified in the solution to the two pup matching problem.

As we already noted, a two pup problem with a single origin reduces to a Directed Steiner Network Problem. For single origin pup matching with many destinations, the heuristic of feasibly loading the optimal directed Steiner tree that connects the common origin to each destination node might seem an attractive extension of the result for two pups. Example 3 and Figure 8 illustrate that the Steiner tree does not necessarily define the optimal solution for single origin problems with more than two pups. Furthermore, Lemma 2 implies that the Matching Approximation provides a lower bound on the best tree solution to the single origin problem.

Example 3 *In the network of Figure 8, suppose we need to route pups A, B, C, and D from node 1 to nodes 2, 3, and 4 as indicated. The optimal solution loads a single cab on each arc for a cost of 6. A tree solution must load two cabs on either arc 1-2 or arc 1-3 and costs at least 7.*

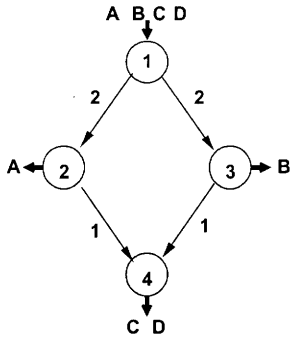

Figure 8: The optimal directed Steiner tree connecting destinations 2, 3, and 4 to origin 1 does not yield the optimal Pup Matching solution.

Lemma 2 *If the pup paths of the single origin problem form a tree, then some routing that pairs each pup with no more than one other pup is optimal given the pup paths.*

Proof. Given any tree forming paths, we can construct the required pairing by first pairing the two pups whose paths from the source node share the greatest number of arcs, then pairing the two remaining pups whose paths share the greatest number of arcs, and so forth. Suppose this routing strategy assigns pups A and B to traverse the same arc singly. Since there is only one path from the source node to each node of the tree, pup A and its pair, if any, and pup B and its pair, if any, share fewer arcs than pups A and B. So, the routing procedure would pair A and B. Consequently, the number of cabs loaded on any arc with a total flow f (which is fixed in the tree) equals $\lceil \frac{f}{2} \rceil$ and so is minimal. □

The Matching Approximation finds a best solution that pairs each pup with at most one other pup and might create a nontree solution, so we have the following result.

Corollary 4 *The matching heuristic provides a lower bound on the best tree solution to the single origin problem.*

Finally, we show that the Matching Approximation (MA) has an absolute performance ratio of 2. If A is an approximation algorithm for a minimization problem π, $A(I)$ is the solution value returned by A on instance $I \in \pi$, and OPT(I) is the optimal value to instance I, then the *performance ratio* of A on the instance I is defined as $R_A(I) = \frac{A(I)}{OPT(I)}$. The *absolute performance ratio* of A is defined as $r_A = \inf\{r | R_A(I) \leq r, \forall I \in \pi\}$.

Theorem 5 *The absolute performance ratio r_{MA} of the Matching Approximation is 2 for both Pup Matching and its NLP formulation.*

Proof. The ratio is no greater than 2 because the Matching Approximation can do no worse than routing each pup singly on its shortest origin-destination path. Specifically, if m_i is the cost of the ith match of the approximation, d_j is the length of the shortest origin-destination path of pup j, and d'_j is the origin-destination path length of pup j in an optimal solution, then for any instance I of Pup Matching, $MA(I) = \sum_i m_i \leq \sum_{j \leq K} d_j \leq \sum_{j \leq K} d'_j \leq 2OPT(I)$.

On the other hand, Figure 9 depicts a sequence of instances with a limiting performance ratio of 2. Instance n consists of $n+1$ nodes and $n+1$ arcs connected as in the figure, as well as n pups. For $i = 1, 2, \ldots (n-1)$, nodes i and $i+1$ form an origin-destination pair for one pup, and nodes 1 and n are the origin-destination pair for the final pup. The optimal solution routes each pup along its unique O-D path among nodes $1, 2, \ldots n$, pairs the final pup with each of the other $n-1$ pups over a single arc, and achieves a cost of $n-1$ since it uses only one cab on each arc. The pairing restriction, however, blinds the Matching Approximation to the efficiency of sending the last pup along nodes $1, 2, \ldots n$. It sends the final pup via node A, routes all pups singly, and loads a single cab on each arc for a total cost of $2n-4$. □

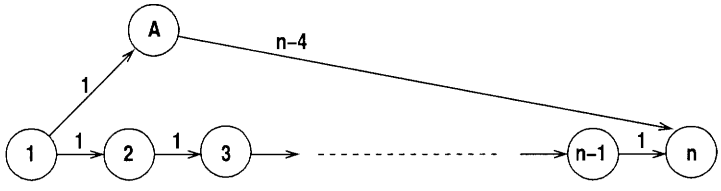

Figure 9: Instance n of a sequence with an infimum Pup Matching performance ratio of 2.

3.2.2 Shortest Path Heuristics

Pup Matching heuristics based on successive shortest path calculations might seem an intuitive approach to the problem. Perhaps the simplest strategy is to route each pup on its shortest path with arc lengths given by facility loading costs. This strategy is equivalent to solving the LP relaxation and rounding up the number of cabs loaded on each arc. Alternatively, we might choose a pup, route it on its shortest path, modify the arc costs to reflect marginal costs, and repeat the procedure until we have

routed all of the pups. If the first pup were routed on some arc, the marginal cost of that arc to the second pup would be 0. A third option combines the preceding two by routing a first subset of pups pups according to loading costs and the rest according to marginal costs. Delayed reduction of marginal costs might allow subsequent pups to better exploit unused capacity. Having determined the $O-D$ routes, we then set the cab capacity on each arc to the integer round up of the flow divided by 2.

Clearly none of these procedures is optimal. Furthermore, the three heuristics do not necessarily output feasible solutions to Pup Matching since each might produce a waiting ring. Each would generate the three node ring of Example 1. However, the heuristics generate feasible solutions to the NLP formulation of Pup Matching and yield 2-approximations for that formulation.

Proposition 1 *Each of the three successive shortest path heuristics provides a 2-approximation for the NLP formulation of the Pup Matching Problem.*

Proof. The cost of each heuristic never exceeds the sum of the shortest $O-D$ paths and half this sum is a lower bound on the optimal solution. So, the heuristics have an absolute performance ratio no worse than 2.

On the other hand, the performance ratio of each heuristic is $\frac{2}{1+\epsilon}$ on the family of instances (parameterized by ϵ) illustrated in Figure 10. Pup A is to be routed from node 1 to node 6, and pup B is to be routed from node 4 to node 6. Each heuristic routes pup A on path 1-2-6 and pup B on path 4-5-6, independently of the routing order, for a cost of 4. The optimal solution routes both pups to node 3 then sends them together to node 6 for a cost of $2+2\epsilon$. □

Epstein [10] described two shortest path based heuristics for a two facility version of the NLP on an undirected graph that he refers to as Edge Rounding and Path Rounding. The former is equivalent to our first shortest path heuristic of routing on shortest paths and rounding up. Epstein outlined instances illustrating that Edge Rounding and Path Rounding have absolute performance ratios equal to the capacity

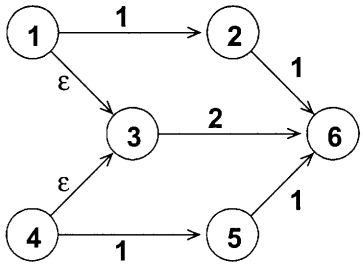

Figure 10: Pup A is to be routed from 1 to 6 and pup B from 4 to 6. The performance ratio of each successive shortest path heuristic is $\frac{2}{1+\epsilon}$.

of the larger facility.

3.3 Valid Inequalities

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

Cutset inequalities (see Magnanti, Mirchandani, and Vachani [14], Barahona [4], or Bienstock and Günlük [7]) bound the capacity loaded across a cut to accommodate the flow that must cross the cut. For Pup Matching, the inequalities assume the following relatively simple form:

$$\sum_{i \in S, j \notin S} z_{ij} \geq \left\lceil \frac{D_S}{2} \right\rceil, \forall S \subset N. \tag{6}$$

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 the arcs leaving 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, as in Balakrishnan, Magnanti, Sokol, and Wang [3], we append the inequalities (one for inflow, one for outflow) for each cut defined by a single node and then iteratively calculate Gomory-Hu trees (see, for example, [1]) to identify other promising cuts.

A residual capacity inequality (see Magnanti, Mirchandani, and Vachani [13], [14]) constrains the loading requirement on a single arc (i, j), with one inequality defined for every commodity subset on every arc. For the Network Loading formulation of Pup Matching, the residual capacity inequalities reduce to:

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

for an odd cardinality subset L of pups. The arc capacity inequalities (3) imply the residual capacity constraints for even cardinality subsets. For a single commodity k, the inequality reduces to the logical condition that flow requires capacity, $z_{ij} \geq f_{ij}^k$.

Atamtürk and Rajan [2] have shown how to separate the residual capacity inequalities for a single arc of a Network Loading Problem with q commodities and with facilities of an arbitrary capacity in $\mathcal{O}(q)$ time. We can also separate the residual capacity inequalities for Pup Matching in $q \log q$ 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 3 *A given fractional solution violates a 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 $\mid L \mid$.*

Our solution procedure checks the inequalities identified by the $q \log q$ routine. Given

that the generation of cutting planes typically accounted for a small fraction of overall solution time on larger instances, implementing the routine of Atamtürk and Rajan seems unnecessary.

Although the cutset and residual capacity inequalities improve the lower bounds, they did not lead to efficient solutions of all the city blocks test problems, including Example 2. In trying to prove optimality of the Example 2 solution shown 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 might be even or odd. However, if the net demand at a node is odd, then its total inflow or total outflow must be odd, and the node must be incident to at least one unit of unused capacity, half 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.

Proposition 2 *The following odd flow inequalities are valid for the NLP formulation of Pup Matching for each node $i \in N$ with odd net demand:*

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

In this expression, A_i denotes the set of arcs (incoming and outgoing) incident to node i.

The odd flow inequalities are a special case of the generalized cutset inequalities that Chopra, Gilboa, and Sastry [9] introduced in the context of the single facility, single O-D pair NLP with both flow and loading costs. Atamtürk showed that the

generalized cutset inequalities yield the convex hull of the NLP variation of shipping a fixed amount of demand across a single directed cut. Our solution procedure appends odd flow inequalities corresponding to only single nodes, though the same logic applies to any subset of nodes with odd net demand.

We show in the appendix that under strong connectivity conditions, the odd flow inequalities define facets of the convex hull of feasible solutions to the NLP formulation.

Theorem 6 *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 convex hull of feasible solutions to the NLP formulation of Pup Matching.*

Proof. See the appendix. □

4 Computational Results

Our solution procedure first finds a lower bound by tightening the LP relaxation of the NLP formulation by adding cutset, residual capacity, and odd flow inequalities, in that order. As mentioned earlier, we use Gomory-Hu calculations to identify interesting cutset inequalities. We exactly separate and append residual capacity inequalities until the bound improvement falls below a threshold, and we append all odd flow inequalities since they number at most the cardinality of the node set. We then obtain an upper bound and initial feasible solution by running all four heuristics and retaining the best value. Finally, we call the CPLEX branch and bound routine.

Figure 11 summarizes the results of our solution procedure on five city blocks problems. Problem 7i is the city blocks problem of Example 2, and 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.

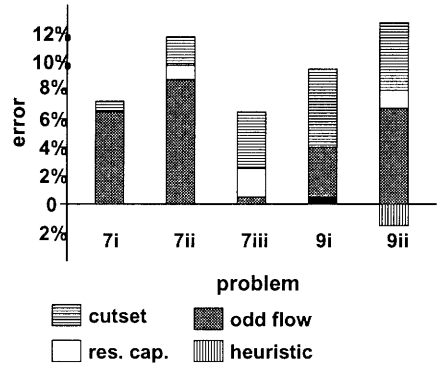

Figure 11: Results of the branch and bound procedure on city blocks problems. The portion of the graph above the zero line depicts lower bound improvement from sequential application of the three cutting plane families (in the order cutset, residual capacity, and odd flow), and the portion below the zero line indicates the error of the best heuristic.

We also applied the solution procedure to thirty 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. 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 gaps reflect relative differences between lower and upper bounds.) 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%.

5 Conclusions

We have investigated four heuristic methods and a cutting plane based branch and bound procedure for solving an Network Loading formulation of the Pup Matching Problem. Among the more realistic test problems that we solved 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.

Despite the apparent practical success of the heuristics, we would consider an approximation algorithm with a bound less than 2 a significant addition to the results we have presented. The Matching Approximation provides a 2-approximation for Pup

Matching, each of the three shortest path based heuristic provides a 2-approximation for the incomplete NLP formulation, and examples show that the ratio of 2 is tight for all four. A ratio of 2 is often readily achieved and seems especially natural for this problem since it coincides with the towing capacity of each cab. Consequently, a tighter algorithm would likely reflect new insight.

Even though the heuristic methods were able to generate good feasible solutions, because of weak linear programming lower bounds, a default implementation of branch and bound was unable to 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 proved very effective. They permitted us to solve in seconds city blocks problems that we were previously unable to solve with days of computation.

Although these cuts are a special case of the generalized cutset inequalities Chopra, Gilboa, and Sastry [9] described for a single origin-destination pair NLP variation, we believe the parity interpretation of validity and our facet definition result are new. 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] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc., 1993.

[2] Alper Atamtürk and Deepak Rajan. On splittable and unsplittable flow capacitated network design arc-set polyhedra. *Mathematical Programming*, 92(2):315–333, 2002.

[3] Anataram Balakrishnan, Thomas L. Magnanti, Joel S. Sokol, and Yi Wang. Modeling and solving the single facility line restoration problem. *Operations Research*, 50(4), 2002.

[4] Francisco Barahona. Network design using cut inequalities. *SIAM Journal on Optimization*, 6(3):823–837, 1996.

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

[6] Cynthia Barnhart and H. Donald Ratliff. Modeling intermodal routing. Technical Report COC-91-11, Georgia Institute of Technology, 1991.

[7] Daniel Bienstock and Oktay Günlük. Capacitated network design - polyhedral structure and computation. *INFORMS journal on computing*, 8(3):243–259, 1996.

[8] John M. Bossert. *Modeling and Solving Variations of the Network Loading Problem.* PhD thesis, Massachusetts Institute of Technology, September 2002.

[9] Sunil Chopra, Itzhak Gilboa, and S. Trilochan Sastry. Source sink flows with capacity installation in batches. *Discrete Applied Mathematics*, 85(3):165–192, 1998.

[10] Rafael Epstein. *Linear Programming and Capacitated Network Loading.* PhD thesis, Massachusetts Institute of Technology, February 1998.

[11] 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*, pages 1–9, 1980.

[12] Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. The point-to-point delivery and connection problems: complexity and algorithms. *Discrete Applied Mathematics*, 36:267–292, 1992.

[13] Thomas L. Magnanti, Prakash Mirchandani, and Rita Vachani. The convex hull of two core capacitated network design problems. *Mathematical programming*, 60(2):233–250, 1993.

[14] Thomas L. Magnanti, Prakash Mirchandani, and Rita Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations research*, 43(1):142–157, 1995.

[15] Pawel Winter. Steiner problem in networks: a survey. *Networks*, 17(2):129–167, 1987.

A Facet Definition Proof

Theorem 7 *If $G = (N, A)$ is strongly connected (contains a directed path between each pair of nodes), the total net demand d_i 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 convex hull of feasible solutions to the NLP formulation of Pup Matching.*

Proof.

Validity.
The odd flow inequality forces a solution to include at least 1 unit of unused capacity, or, equivalently, half a cab's worth, on the arcs incident to node i. A solution can fully utilize the capacity on a set of arcs only if the flow on every such arc is even. However, the flow on at least one arc incident to node i must be odd because the odd demand d_i forces either the total inflow or the total outflow to be odd.

More formally, we can derive the odd flow inequality as a rank 1 Gomory-Chvával cut:

Addition of the capacity constraints for the arcs A_i incident to node i yields:

$$2 \sum_{a \in A_i} z_a - \sum_{k \leq K} \sum_{a \in A_i} f_a^k \geq 0.$$

Substitution via the node i flow balance constraints (2), summed over all commodities, yields:

$$2 \sum_{a \in A_i} z_a - 2 \sum_{k \leq K} \sum_{\text{a leaving i}} f_a^k \geq \mid d_i \mid .$$

Division by 2 and rounding yields:

$$\sum_{a \in A_i} z_a - \sum_{k \leq K} \sum_{\text{a leaving i}} f_a^k \geq \left\lceil \frac{\mid d_i \mid}{2} \right\rceil .$$

Subtracting half of the node i flow balance yields the odd flow inequality. We take the absolute value of d_i since the net demand might be negative, that is, a supply.

Face Definition.
We first show that the odd flow inequality can hold at equality. The clique assumption permits us to modify any flow satisfying the flow balance constraints to some

other solution satisfying the properties that some arc (j,i) carries all node i inflow, and, similarly, some arc (i,l) carries all node i outflow. Since the demand d_i is odd, either $\sum_k f_{ji}^k$ or $\sum_k f_{il}^k$ is odd. The odd flow inequality then holds at equality if $z_a = \left\lceil \frac{\sum_k f_a^k}{2} \right\rceil, \forall a \in A$.
Also, for some feasible solution, the odd flow inequality does not hold at equality, because we can add excess loadings to any feasible solution to obtain another feasible solution. Therefore, the odd flow inequality is a nonempty proper face of the convex hull of feasible solutions.

Facet Definition.
Let P be the set of feasible solutions to the network loading problem, and let $L = \{(z,f) \in \text{conv}(P) \mid \text{ the odd flow inequality holds at equality}\}$. Suppose some other inequality $\beta z + \gamma f \geq \delta$ (**) satisfies the property that
$L \subseteq \{(z,f) \mid (**) \text{ holds at equality}\}$. We will show that (**) is a linear combination of the odd flow inequality and the flow balance equalities, implying that $\dim(L) = \dim(\text{conv}(P))$ - 1 and that L is a facet of conv(P).

(a) $\beta_a = 0$ for all arcs a not incident to node i, because for any $(z,f) \in L$, the solution given by increasing z_a by 1 is also in L.

(b) $\beta_a = \beta$ for some constant β for all a incident to node i.
Let (z_0, f_0) be the feasible solution described previously, with all the flow into node i via some other node j, all the flow from node i via some node l, and $z_a = \left\lceil \frac{\sum_k f_a^k}{2} \right\rceil, \forall a \in A$. Modify (z_0, f_0) by sending one additional unit of flow of some commodity k around the cycle $(j,i) \rightarrow (i,l) \rightarrow (l,j)$ (If $l = j$, ignore the arc (l,j), and if $l \neq j$, arc (l,j) exists by the clique assumption.), and incrementing the loading on either (j,i) or (i,l) and (l,j) if necessary, to maintain feasibility. Call the new solution (z_1, f_1).

Form a third solution (z_2, f_2) by modifying (z_1, f_1) in the same manner. Note that if arc (j,i) capacity is tight in (z_0, f_0), then arc (i,l) capacity is tight in (z_1, f_1), and vice versa, and that (z_0, f_0), (z_1, f_1), and (z_2, f_2) are all in L. Assume without loss of generality that arc (j,i) is tight in (z_0, f_0). Then,

$$(\beta z_1 + \gamma f_1) - (\beta z_0 + \gamma f_0) = \beta_{ji} + \gamma_{ji} + \gamma_{il} + \gamma_{lj} = 0. (\gamma_{lj} \text{ is irrelevant if } l = j).$$

Similarly,

$$(\beta z_2 + \gamma f_2) - (\beta z_1 + \gamma f_1) = \beta_{il} + \gamma_{ji} + \gamma_{il} + \gamma_{lj} = 0 \Rightarrow \beta_{ji} = \beta_{il}.$$

Since we chose nodes j and l arbitrarily among adjacent nodes, $\beta_a = \beta$ for all arcs a incident to node i.

(c) $\gamma_a^k = 0, \forall a$ not incident to node i.
Consider again node j, and let $T = (N, A')$ be a directed spanning tree formed by directed paths from this node to each other node. Such a tree exists by the strong connectivity assumption. Furthermore, assume that node i is a leaf of T connected

to the tree by arc (j,i). The clique assumption guarantees that we can reroute any paths through node i via adjacent nodes. We can modify (**) by adding to it flow balance constraints so that $\gamma_a^k = 0, \forall a \in A', \forall k \leq K$. We could prove this claim using induction on the number of nodes.

To show that $\gamma_a^k = 0$ for $a \notin A'$ not incident to node i, first modify the initial solution (z_0, f_0) by adding 1 unit of flow through T for some commodity k to each node $p \neq j, \neq i$ and along each arc of a $p - j$ directed path that does not include node i. Strong connectivity and the clique assumption guarantee that such directed paths exist. Load $\left\lceil \frac{\sum_k f_a^k}{2} \right\rceil + 1$ facilities on all arcs a not incident to i and $\left\lceil \frac{\sum_k f_a^k}{2} \right\rceil$ facilities on all arcs incident to i. Call the resulting solution (z_3, f_3), and note that $(z_3, f_3) \in L$ since the flows on arcs incident to node i are the same as those of (z_0, f_0). Now consider some arc $(r,q) \notin A'$ that is not incident to node i. Incrementing the flow of k on (r,q) and the $j - r$ path of T, and decrementing the flow on the $j - q$ path of T, generates a new solution in L. Since $\gamma_a^k = 0, \forall a \in A', \gamma_{r,q}^k = 0$. Since we chose commodity k and arc (r,q) arbitrarily, the result follows.

(d) $\gamma_{i,l'}^k = -\beta, \forall k \in K$.
Increment the flow of f_0 around the cycle $(l,j) \rightarrow (j,i) \rightarrow (i,l)$ (if $l = j$, ignore the first arc, which does not exist) for some commodity k. One additional loading will be required on either (j,i) or (i,l), and, perhaps, on (l,j). The resulting solution $(z_4, f_4) \in L$. Comparing (z_0, f_0) with (z_4, f_4) yields $\gamma_{j,i}^k + \gamma_{i,l}^k + \beta = 0$
$\Rightarrow \gamma_{i,l}^k = -\beta$, since (j,i) is in the tree T.
Now suppose the network contains some other adjacent node l', and modify (z_4, f_4) by sending 2 additional units of k around the cycle $(l,j) \rightarrow (j,i) \rightarrow (i,l') \rightarrow (l',l)$ (again, ignore (l,j) if $l = j$, and if $l' = j$, the argument still holds). Add 1 loading to each of $(l,j), (j,i), (i,l')$, and (l',l) to create $(z_5, f_5) \in L$. Comparing (z_5, f_5) with (z_4, f_4) yields $2\gamma_{j,i}^k + 2\gamma_{i,l'}^k + 2\beta = 0$
$\Rightarrow \gamma_{i,l'}^k = -\beta$.
$\Rightarrow \gamma_a^k = -\beta, \forall a$, with i as tail node, for all commodities.

(e) $\gamma_{j',i}^k = 0, \forall j', \forall k \in K$.
Consider some node $j' \neq j$ adjacent to node i. Modify (z_0, f_0) by adding 2 units of flow of commodity k around the cycle $(l,j') \rightarrow (j',i) \rightarrow (i,l)$ (if $l = j'$, ignore (l,j'), analogous to before), and by adding 1 loading to each of $(l,j'), (j',i)$ and (i,l), to create $(z_6, f_6) \in L$. Comparing (z_6, f_6) and (z_0, f_0) yields $2\gamma_{j',i}^k + 2\gamma_{i,l}^k + 2\beta = 0$
$\Rightarrow \gamma_{j',i}^k = 0$.
So, $\gamma_{j',i}^k = 0$ for all arcs with head node i, for all commodities k.

(f) Summary
$\beta_a = 0$, for all arcs a not incident to node i,
$\beta_a = \beta$, for all arcs a incident to node i, for some constant β,
$\gamma_{j,i}^k = 0, \forall j \in N, \neq i, \forall k \leq K$,
$\gamma_{i,l}^k = -\beta, \forall l \in N, \neq i, \forall k \leq K$,
$\gamma_{r,q}^k = 0, \forall (r,q)$ not incident to i, $\forall k \leq K$.
Using the flow balance constraints on node i, we can convert γ so that
$\gamma_{j,i}^k = -\frac{1}{2}\beta$,

$\gamma^k_{i,l} = -\frac{1}{2}\beta$,
$\gamma^k_{r,q} = 0$. $\square$

The clique assumption is not necessary. The proof still holds if we assume only a directed path not including node i between any nodes j and l adjacent to node i.