Minimum Cost Capacity Installation for Multicommodity Network Flows*

Daniel Bienstock

Dept. of IEOR, Columbia University and Bellcore

Sunil Chopra

J.L. Kellogg Graduate School of Management, Northwestern University

Oktay Günlük

CORE, Université Catholique de Louvain

Chih-Yang Tsai

State University of New York at New Paltz

July 1995 (Revised)

Abstract

Consider a directed graph $G = (V, A)$, and a set of traffic demands to be shipped between pairs of nodes in V. Capacity has to be installed on the edges of this graph (in integer multiples of a base unit) so that traffic can be routed . In this paper we consider the problem of minimum cost installation of capacity on the arcs to ensure that the required demands can be shipped simultaneously between node pairs.

We study two different approaches for solving problems of this type. The first one is based on the idea of metric inequalities (see Onaga and Kakusho[1971]), and uses a formulation with only $|A|$ variables. The second uses an aggregated multicommodity flow formulation and has $|V| \cdot |A|$ variables. We first describe two classes of strong valid inequalities and use them to obtain a complete polyhedral description of the associated polyhedron for the complete graph on 3 nodes. Next we explain our solution methods for both of the approaches in detail and present computational results. Our computational experience shows that the two formulations are comparable and yield effective algorithms for solving real-life problems.

*The research of the first and third authors was partially supported by a Presidential Young Investigator Award, by NSF grant NCR-9301751, and by a post-doctoral fellowship at CORE

1 Problem Definition, Formulations and Complexity

In this paper, we study the following optimization problem: Consider a directed graph $G = (V, A)$ with node set V, arc set A, and a set of demands $d_{i,j}$, each specified by an ordered pair of nodes $[i, j]$, $i, j \in V$ and $i \neq j$. Each pair $[i, j]$ corresponds to a commodity to be sent from source node i to destination node j using arcs in A. Let $K = \{1, \ldots, |K|\}$ be the set of indices of the demand pairs (commodities). For the k^{th} demand pair (commodity) $[i, j]$, $k_s = i$ denotes the source node and $k_d = j$ indicates the destination node. The demand for commodity k (with source node i and destination node j) is denoted either as $d_{i,j}$ or as d^k, and may be fractional. It is also possible to group commodities by source (or destination) and we will discuss this later. Capacities are installed on the arcs in integral units, and c_a denotes the cost of obtaining one unit of capacity on arc a. The objective is to obtain a minimum cost installation of capacities to ensure that all commodities can be shipped simultaneously. We refer to this problem as the *Minimum Cost Capacity Installation* (MCCI) problem. Applications of MCCI are commonly seen in the design of telecommunication and distribution networks. Closely related problems have been studied in [1], [5], [6], and [9].

The objective of this study is to find an efficient algorithm that either solves the problem to optimality or obtains a tight upper bound in reasonable amount of computational time and memory.

The paper is organized as follows: Later in this section we introduce the standard mixed-integer *multicommodity flow formulation* for MCCI and then we transform this formulation to a lower dimensional space by projecting out all continuous variables. The resulting *capacity formulation* uses only $|A|$ integer variables corresponding to the capacities installed on arcs of the network. The transformation uses the so-called *rounded metric inequalities*, obtained by rounding the right-hand-side of the metric inequalities due to Onaga and Kakusho [13]. We also show the problem to be strongly $\mathcal{NP}$-*hard* on general graphs, and $\mathcal{NP}$-*hard* even on series parallel graphs.

In Section 2 we identify a special class of rounded metric inequalities called partition inequalities. In general there is no bound for the coefficients in the rounded metric inequalities. We prove that for the special case of partition inequalities, the resulting coefficients can be bounded by the size of the partition. As a result, for a fixed partition, one can identify violated partition inequalities in polynomial time. In Section 3 we introduce the total capacity inequality and prove that partition inequalities along with the total capacity inequality completely define the polyhedron for the complete graph on 3 nodes.

In Sections 4 and 5 we describe our approach to solving the problem via the capacity and multicommodity formulations, respectively. In Section 6 computational results are presented for both formulations.

1.1 The Multicommodity Formulation:

We next present the standard multicommodity formulation for the MCCI problem. In this formulation, we use y_{ij} to denote the capacity purchased on arc (i, j) and x_{ij}^k as the amount of the k^{th} commodity flowing through arc (i, j). The multicommodity formulation can be written as follows, where we denote the demand for commodity k at node i by d_i^k.

$$
\textbf{(MCF)} \qquad
\begin{array}{rlll}
\text{Min} & \sum_{(i,j)\in A} c_{ij} y_{ij} & & \\
\text{st.} & \sum_j x_{ji}^k - \sum_j x_{ij}^k = d_i^k & \forall k \in K,\ i = k_d & (1) \\
& \sum_j x_{ji}^k - \sum_j x_{ij}^k = 0 & \forall k \in K, i \notin \{k_s, k_d\} & (2) \\
& \sum_{k\in K} x_{ij}^k \leq y_{ij} & \forall (i,j) \in A & (3) \\
& y_{ij} \in N^+ & \forall (i,j) \in A & (4) \\
& x_{ij}^k \geq 0 & \forall (i,j) \in A, \quad \forall k \in K &
\end{array}
$$

In this formulation, there are $|A|$ (integer) capacity variables and $|A||K|$ (continuous) flow variables. The objective is to minimize the total cost of installing the required capacity on the arcs. The first three constraints are flow constraints for each commodity k, requiring that exactly d^k units of commodity k be sent from k_s to k_d. Constraint (3) ensures that the total flow on each arc cannot exceed the capacity purchased on the arc. The number of variables in this formulation is bounded by $\mathcal{O}(|A||K|)$ and the number of constraints is of order $(|V|^2)$ if $|V| \geq |K|$ and of order $(|V||K|)$ otherwise.

We note here that there are many possible choices for what we call "commodities." For example, one could declare that each source-destination demand gives rise to a separate commodity. Based on previous experience, we prefer to aggregate commodities by source, i.e. all demands with a common source constitute a commodity (and thus we have at most $|V|$ commodities). This leads to a significant reduction in problem size while retaining the value of the LP-relaxation. Even though one might generate stronger cutting planes for the disaggregated formulation, we feel that the savings inherent in using the aggregated formulation make it (computationally) much more attractive. See [15], [4] for related discussion. Our approach for solving the multicommodity formulation uses the aggregated formulation, whereas in the discussion of the capacity formulation we will implicitly refer to the disaggregated formulation.

1.2 Metric Inequalities and The Capacity Formulation

It is also possible to reformulate the problem using (an exponential number of) path variables as follows. For each commodity k let $\{P_k^q, q \in \{1, ..., Q_k\}\}$ be the set of all directed paths from k_s to k_d. For each path P_k^q define a variable x_k^q to represent the flow of commodity k on path P_k^q. In formulation (PTH), $\mathbf{A}$ is a 0-1 matrix whose rows are indexed by arcs and columns by paths P_k^q. An entry is 1 if the arc is contained in the path corresponding to the column. Let $\mathbf{B}$ be a 0-1 matrix with a row for each commodity k and a column for each path P_k^q. An entry of $\mathbf{B}$ is 1 in the row corresponding to commodity k and the column corresponding to a path if the path is from k_s to k_d. $\mathbf{d}$ is a column vector with entry d_k corresponding to commodity k.

$$
(\mathbf{PTH}) \qquad
\begin{array}{lll}
\text{Min} & \mathbf{cy} & \\
\text{st.} & \mathbf{Ax} - \mathbf{Iy} & \leq \mathbf{0} \qquad (5) \\
 & -\mathbf{Bx} & = -\mathbf{d} \qquad (6) \\
 & \mathbf{x},\ \mathbf{y} \geq \mathbf{0}, & \mathbf{y}\ \textit{integer}. \qquad (7)
\end{array}
$$

There are $|A|+|K|$ rows in this formulation and the number of variables can be exponential.

One can verify that the two formulations (MCF) and (PTH) are identical in terms of the strength of the LP-relaxation, i.e., for each vector in the LP-relaxation of (MCF), there is a corresponding vector (with the same objective function value) in the LP-relaxation of (PTH) and vice versa.

Next, we describe a formulation obtained by projecting out the continuous variables from the (PTH) formulation. Let $P(PTH)$ be the convex hull defined by the feasible solutions to formulation (PTH) and $LP(PTH)$ be the polyhedron determined by the LP relaxation of formulation (PTH). Let $P(C)$ be the projection of $P(PTH)$ down to the space $\mathcal{R}^{|A|}$ by projecting out path variables x_k^q. Using $P(C)$, we only need $|A|$ capacity variables y_a, $a \in A$ in the formulation defining $P(C)$. The resulting formulation uses the rounded metric inequality introduced later in this section.

Note that a capacity vector $\bar{\mathbf{y}}$ allows a feasible flow of all commodities if and only if $\{\mathbf{x} \in R^{|A|} | \mathbf{Ax} \leq \bar{\mathbf{y}},\ \mathbf{Bx} = \mathbf{d},\ \mathbf{x} \geq \mathbf{0}\} \neq \emptyset$,i.e., there exists a solution to the following system of equations:

$$
(\mathbf{I}) \qquad
\begin{array}{ll}
-\mathbf{Ax} - \mathbf{Is} & = -\bar{\mathbf{y}} \qquad (8) \\
+\mathbf{Bx} + \mathbf{0s} & = \mathbf{d} \qquad (9) \\
\mathbf{x},\ \mathbf{s} & \geq \mathbf{0}.
\end{array}
$$

By Farkas Lemma, system (I) has a solution if and only if the following system is infeasible:

$$\textbf{(II)} \qquad \begin{aligned} -\bar{\mathbf{y}}\mathbf{w} + \mathbf{d}\mathbf{z} &> 0 &(10)\\ -\mathbf{A}^T\mathbf{w} + \mathbf{B}^T\mathbf{z} &\le \mathbf{0} &(11)\\ -\mathbf{I}\mathbf{w} \le \quad & \mathbf{0} &(12)\end{aligned}$$

where $\mathbf{w}$ and $\mathbf{z}$ are dual variables corresponding to row (8) and (9) respectively. To check feasibility of system (II) we solve the following LP

$$\textbf{(III)} \qquad \begin{aligned} \text{Min} \quad & \bar{\mathbf{y}}\mathbf{w} - \mathbf{d}\mathbf{z} \\ & \mathbf{A}^T\mathbf{w} - \mathbf{B}^T\mathbf{z} \ge \mathbf{0} \\ & \mathbf{w} \qquad\quad \ge \mathbf{0}. \end{aligned} \tag{13}$$

If the capacity vector $\bar{\mathbf{y}}$ admits a feasible flow in (I), the optimal solution to (III) is non-negative. If $\bar{\mathbf{y}}$ admits no feasible flow in (I) there exists an extreme direction $(\mathbf{w}^*, \mathbf{z}^*)$ for problem (III) with $\mathbf{w}^*\bar{\mathbf{y}} - \mathbf{d}\mathbf{z}^* < 0$. In this case the inequality

$$\mathbf{w}^*\mathbf{y} \ge \mathbf{d}\mathbf{z}^* \tag{14}$$

separates $\bar{\mathbf{y}}$ from $P(C)$. We refer to the inequality (14) as the *metric inequality.* (Note that for any extreme direction $(\mathbf{w}, \mathbf{z})$ of (13), z corresponds to the shortest path lengths using w as edge weights.) These inequalities were first described by Onaga and Kakusho(1971). Since A and B are rational, we can restrict attention to the case where $\mathbf{w}^*$ has only integer values. In this case inequality (14) can be further strengthened to the *rounded metric inequality*

$$\mathbf{w}^*\mathbf{y} \ge \lceil \mathbf{d}\mathbf{z}^* \rceil. \tag{15}$$

¿From the above results it follows that $P(C) = \{\mathbf{y} \in N^+ \mid \mathbf{y}$ satisfies (15) for all integral weights w^* coming from III $\}$. MCCI can thus be formulated as

$$\text{Min } \{ \sum_{(i,j)\in A} c_{ij}y_{ij} | \text{ s.t. } y \in N^+, \text{ y satisfies (15) for all integral weights } w^* \text{ coming from III } \}.$$

This formulation is referred to as the *capacity formulation.* We define $LP(C)$ as the continuous relaxation of $P(C)$.

1.3 Complexity

It can be shown that the MCCI problem on general graphs is strongly $\mathcal{NP}$*-hard*, as it contains Set Covering as a special case. Here, we will show that MCCI on series-parallel graphs is as hard as the Knapsack problem.

Claim 1.1 *The minimum cost capacity installation problem on series-parallel graphs is $\mathcal{NP}$-hard.*

Proof: Given the knapsack problem $max\{\sum_{i=1}^{n} w_i x_i \mid \sum_{i=1}^{n} f_i x_i \leq b, x_i \in Z\}$ where $f_i < 1$ for all i and $b < 1$, construct the directed graph $G' = (V', A')$ where $V' = \{s, 1, 2, \cdots, n, n+1\}$, $A' = \{(s,i)|i = 1, \cdots, n+1\} \cup \{(n+1, i)|i = 1, \cdots, n\}$, $d_{s,i} = f_i$ for $i \in \{1, \cdots, n\}$ and $d_{s,n+1} = 1-b$. Set $c_{s,i} = w_a$ for all $\{(s,i)|i = 1, \cdots, n\}$, $c_a = 0$ for $a \in \{(n+1, i)|i = 1, \cdots, n\}$ and $c_{s,n+1} = M$ where M is a number greater than $\sum_{i=1}^{n} w_n$. A commodity is defined for each of the nodes $1, ..., n+1$. The resulting network is shown in Figure 1.

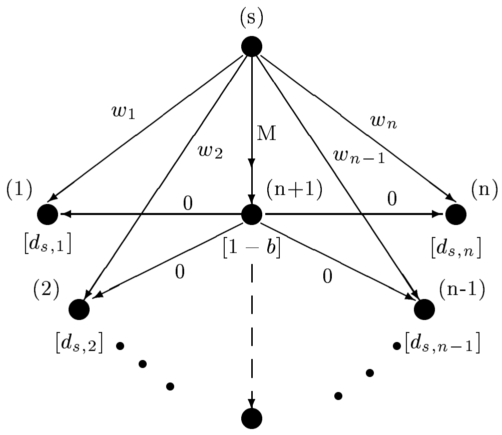

Figure 1: Graph $G' = (V', A')$ of Claim 1.1

Note that there is only one path from the source, s, to node $n+1$. Therefore, one unit of capacity on arc $(s, n+1)$ must be purchased. After sending $1-b$ units designated for node $n+1$, there is still b units of residual capacity left on arc $(s, n+1)$. For any node $i \in \{1, \ldots, n\}$, there are two paths from s to i, one using the arc $(s, n+1)$ and the other direct.

Given the optimal solution to this MCCI problem one obtains the optimal solution to the knapsack problem by setting $x_i = 1$ in the knapsack problem if $d_{s,i}$ (the entire flow of commodity i) is sent through arc $(s, n+1)$. Since this transformation is linear, it follows that MCCI is also $\mathcal{NP}$-*hard*. □

2 Partition Inequalities

In general it is not known if the coefficients for the rounded metric inequalities (16) can be bounded by an integer related to the size of the graph G. In this section we identify a class of rounded metric inequalities (partition inequalities) for which this is the case. As a result, for a fixed partition, one can solve the separation problem for partition inequalities in polynomial time. This is not true for general rounded metric inequalities. An explicit description of these inequalities also allows for the possibility of identifying violated inequalities without having to solve problem (III). This is valuable since the solution to problem (III) can be fairly time consuming.

Assume that G is a complete directed graph, i.e., for any pair of nodes u and v, the arc $(u, v) \in A$. Let $\pi = (V_i, i = 1, \cdots, l(\pi))$ be any partition of the node set V. Let $\alpha : \{1, \cdots, l(\pi)\} \to \{1, \cdots, l(\pi)\}$ be any permutation function. $\alpha(\pi)$ is a partition of the node set V where each subset is ordered as defined by α. Define

$$A(\alpha(\pi)) = \{(u, v) \in A \mid u \in V_i, v \in V_j, i \neq j, \alpha(i) < \alpha(j)\}$$

to be the set of arcs directed from lower numbered subsets to higher numbered subsets. For each arc $a \in A - A(\alpha(\pi))$ define $w_a^* = 0$. For each arc $a \in A(\alpha(\pi))$ assign a non-negative integer weight w_a^*. For each commodity k, let l^k be the length of the shortest path from the source k_s to the destination k_d using the arc weights w^* described above. Notice that if $k_s \in V_i$ and $k_d \in V_j$ where $\alpha(i) > \alpha(j)$, then $l^k = 0$. The inequality

$$\sum_{a \in A(\alpha(\pi))} w_a^* y_a \geq \lceil \sum_{k=1}^{|K|} l^k d^k \rceil \tag{16}$$

is a valid inequality for $P(C)$, since it is a special case of the rounded metric inequality (15). The inequality (16) is referred to as the *partition inequality*.

In the following discussion we show that for partition inequalities the coefficients can be limited by the size of the partition. Given a partition π, we identify a minimal set of integer weights that need to be considered to define all the required inequalities (16) from this partition. The following assumptions are made regarding the weights w^* and the partition π.

(i) Define $\pi^* = \{V_i^*, i = 1, ..., l(\pi^*)\}$ to be a *coarsening* of π if $l(\pi^*) < l(\pi)$ and for each $1 \leq i \leq l(\pi)$ we have $V_i \subseteq V_j^*$ for some $j \in \{1, ..., l(\pi^*)\}$. We assume that the inequality (16) cannot be defined by a coarsening of partition π.

(ii) The weights w^* are such that there exists a vector y^* that violates inequality (16) but satisfies all the partition inequalities obtained by using the partition π and weights w where $w \neq w^*, w \leq w^*$.

Note that we are only interested in facet defining partition inequalities and every facet defining partition inequality satisfies assumptions (i) and (ii). For ease of notation we also assume that $\alpha(i) = i$ for the rest of this section. We assume that the weights w^* are scaled such that w^* is integral.

Claim 2.1 *Each arc a with $w^*_a \geq 1$, is contained in at least one path defining l^k (i.e., a shortest path), for some k.*

Proof: To the contrary assume that $w^*_{a^*} \geq 1$ but arc a^* is not contained in any shortest path for any commodity. Define the vector w' where $w'_a = w^*_a$ for $a \neq a^*$ and $w'_{a^*} = w^*_{a^*} - 1$. Note that the partition inequality defined by the weights w' has the same right hand side as the one defined by the weights w^* since the lengths of the shortest paths l^k are unchanged. Thus the weights w^* contradict assumption (ii). The result thus follows. □

Claim 2.2 *Consider arcs $a_i = (s_i, t_i), s_i \in V_j, t_i \in V_k, i = 1, 2, j \neq k$. Then $w^*_{a_1} = w^*_{a_2}$.*

Proof: To the contrary assume that $w^*_{a_1} < w^*_{a_2}$. Clearly the arc a_2 is not used in any shortest path defining l^k, for any k, contradicting Claim 2.1. The result thus follows. □

Claim 2.3 *Consider arcs $a_i = (s_i, t_i)$, $i = 1, 2$, where $s_i \in V_j, i = 1, 2, t_1 \in V_k, t_2 \in V_r$ for $j < k < r$. Then $w^*_{a_1} \leq w^*_{a_2}$.*

Proof: To the contrary assume that $w^*_{a_1} > w^*_{a_2}$. Let $a' = (t_2, t_1)$. Note that $w^*_{a'} = 0$. Thus the shortest path from s_1 to t_1 can be at most $w^*_{a_2}$. This implies that the arc a_1 is not used in any of the shortest paths defining l^k for any k, contradicting Claim 2.1. The result thus follows. □

Claim 2.4 *Consider arcs $a_i = (s_i, t_i), i = 1, 2, 3$ where $\{s_1, s_2\} \subseteq V_j$, $\{s_3, t_2\} \subseteq V_k$, $\{t_1, t_3\} \subseteq V_r$, $j < k < r$. Then $w^*_{a_1} \leq w^*_{a_2} + w^*_{a_3}$.*

Proof: To the contrary assume that $w^*_{a_1} > w^*_{a_2} + w^*_{a_3}$. In this case no shortest path defining l^k, for any k, contains the arc a_1. This contradicts Claim 2.1 proving the result. □

Claim 2.5 *If $a = (s, t)$ with $s \in V_i, t \in V_j, i < j$, then $w^*_a \geq 1$.*

Proof: To the contrary assume that $w^*_a = 0$. By Claims 2.2 and 2.3, if $a' = (s', t')$ for $s' \in V_i$ and $t' \in V_k, i < k < j$ then $w^*_{a'} = 0$. Now consider the case where $\bar{a} = (\bar{s}, \bar{t})$ where $\bar{s} \in V_k, \bar{t} \in V_l$ for $i < k < l < j$. Let $a^* = (\bar{s}, s)$ and $a = (s, \bar{t})$. By assumption $w^*_{a^*} = 0$. Note that $w^*_a = 0$ since $i < l < j$. Using Claim 3.4 we have $w^*_{\bar{a}} \leq w^*_a + w^*_{a^*} = 0$. This shows that $w^*_{\bar{a}} = 0$ for all $\bar{a} = (\bar{s}, \bar{t})$, $\bar{s} \in V_k$, $\bar{t} \in V_l$, $i \leq k < l \leq j$. We can thus define a partition π' where subsets V_i through V_j are merged into a single subset with all other subsets the same as in π. The partition π' with weights w^* defines the same partition inequality as π. Since π' is a coarsening of π, assumption (i) is contradicted. The result thus follows. $\square$

Claim 2.6 *If $a^* = (s^*, t^*)$ where $s^* \in V_i, t^* \in V_{i+1}$, we have $w^*_{a^*} = 1$.*

Proof: To the contrary assume that $w^*_{a^*} \geq 2$. Let $a_j = (s_j, t_j), j = 1, 2$ where $s_1 \in V_k$, $k < i$, $s_2 \in V_i$, $t_1 \in V_{i+1}$, $t_2 \in V_j$, $j > i + 1$. By Claims 2.2, 2.3, and 2.4 and the fact that $w^*_{(s^*,s_1)} = w^*_{(t_2,t^*)} = 0$, we have $w^*_{a_i} \geq 2$. Now consider $\bar{a} = (\bar{s}, \bar{t})$ where $\bar{s} \in V_k$, $k < i$, $\bar{t} \in V_j$. Define $a_3 = (\bar{s}, t^*)$ and $a_4 = (\bar{t}, t^*)$. By claims 2.2 and 2.4 we have

$$2 \leq w^*_{a_3} \leq w^*_{\bar{a}} + w^*_{a_4} = w^*_{\bar{a}}.$$

Define $V^* = \cup_{l=1}^{i} V_l$. Let $A^* = \{a = (s, t) \mid s \in V^*, t \notin V^*\}$. Let K^* be the set of commodities with k_s in V^* and k_d not in V^*.

Given a vector y, define $X_1(y) = \sum_{a \in A^*} y_a$ and $X_2 = \lceil \sum_{k \in K^*} d_k \rceil$. Note that the inequality $X_1(y) \geq X_2$ is also a partition inequality where the arcs in $\bar{A}$ are assigned a weight of 1 with all other arcs getting a weight of 0. By assumption (ii) there exists a vector y^* which violates the partition inequality defined by the weights w^* but satisfies the inequality $X_1(y^*) \geq X_2$ (in fact y^* satisfies all partition inequalities defined by weight vectors smaller than w^*). Define the weight vector w' where $w'_a = w^*_a - 1$ for all arcs a in A^* and $w'_a = w^*_a$ otherwise. Let $\bar{l}_k$ be the shortest path between k_s and k_d with arc weights w'. Note that $\bar{l}_k \geq l_k - 1$ with $\bar{l}_k = l_k - 1$ for k in K^* and $\bar{l}_k = l_k$ for k not in K^*. By assumption (ii) we have

$$\sum_{a \in A} w'_a y^*_a \geq \lceil \sum_{k \in K} \bar{l}_k d_k \rceil$$

since the weight vector w' also defines a partition inequality. Note that

$$\sum_{a \in A} w^*_a y^*_a = \sum_{a \in A} w'_a y^*_a + X_1(y^*)$$

and

$$\lceil \sum_{k \in K} l_k d_k \rceil = \lceil \sum_{k \in K^*} (\bar{l}_k + 1) d_k + \sum_{k \notin K^*} l_k d_k \rceil$$
$$\leq \lceil \sum_{k \in K^*} \bar{l}_k d_k \rceil + \lceil \sum_{k \notin K^*} l_k d_k \rceil + \lceil \sum_{k \in K^*} d_k \rceil \leq \lceil \sum_{k \in K} \bar{l}_k d_k \rceil + X_2.$$

This implies that

$$\sum_{a \in A} w^*_a y^*_a = \sum_{a \in A} w'_a y^*_a + X_1(y^*) \geq \lceil \sum_{k \in K} \bar{l}_k d_k \rceil + X_2 \geq \lceil \sum_{k \in K} l_k d_k \rceil,$$

contradicting our assumption that y^* violates the partition inequality defined by w^*. The result thus follows. $\square$

Using Claims 2.4 and 2.6 we are able to restrict the integers that need to be considered when defining partition inequalities.

Claim 2.7 *For any facet defining partition inequality, if $a = (s,t), s \in V_i, t \in V_j, j > i$. we can restrict attention to weights w^* where $1 \leq w^*_a \leq j - i$.*

Therefore, given a capacity vector $\bar{y}$ and a partition π of fixed size $l(\pi)$, one can identify in polynomial time any violated partition inequality arising from this partition. (Given the partition π, there are $l(\pi)!$ choices of the ordering function $\alpha(\pi)$. For each choice of $\alpha(\pi)$ there are at most $(l(\pi))^{2l(\pi)}$ choices of the coefficients on the arcs by Claim 2.7.) We also note that we cannot obtain such a result for general rounded metric inequalities since we cannot bound the coefficients as in Claim 2.7.

It is hard to identify general conditions under which a weight vector w^* results in a facet defining inequality (15) for a partition π and a permutation α. There is one instance in which this can be done. Given a partition π define

$$d_{ij} = \sum_{k:k_s \in V_i, k_d \in V_j} d_k,$$

and

$$(d_{ij})_f = d_{ij} - \lfloor d_{ij} \rfloor.$$

The result is stated here without proof. The proof uses standard techniques in polyhedral theory (see [10]).

Claim 2.8 *Given a partition $\pi = (V_i, i = 1, ..., k(\pi))$ consider the permutation α where $\alpha(i) = i$. Define the weight vector w^* such that $w^*_a = 1$ for all arcs a in $A(\alpha(\pi))$ and $w^*_a = 0$ for all other arcs. If demands d_k are such that $(d_{ij})_f \geq (k(\pi) - 1)/k(\pi)$, for $i < j$, the corresponding partition inequality is facet defining for $P(C)$.*

Remark: We give an example to show that other weight vectors can also give facet defining partition inequalities. Consider the case where $k(\pi) = 4$ and $(d_{ij})_f = 1/3$ for all $i < j$. Define $w^*_{(s,t)} = 2$ for $s \in V_1, t \in V_4$, $w^*_a = 1$ for all other arcs in $A(\alpha(\pi))$, and $w^*_a = 0$ for the remaining arcs. One can prove that the resulting partition inequality is facet defining, while the partition inequality obtained by setting $w^*_a = 1$ for all $a \in A(\alpha(\pi))$ is not.

3 P(C) for the Complete Graph on 3 Nodes

In general we cannot hope to characterize completely the polyhedron $P(C)$. However we are able to do so for the case where $G = (V, A)$ is the complete directed graph on 3 nodes, i.e., $\mid V \mid= 3$. Note that in this case we can have partition inequalities (16) for partitions of size 2 or 3. In general, we need all the metric inequalities (14) to guarantee feasibility of a capacity vector. We first prove that in the case of $\mid V \mid= 3$ we need only the partition inequalities to guarantee feasibility. We show that the partition inequalities along with a Chvátal-Gomory cut provide a complete description of $P(C)$ for the complete directed graph on 3 nodes.

Let $\mathcal{R} = \{\mathbf{y} | \mathbf{Ay} \geq \mathbf{b},\ \mathbf{y} \in R\}$ where $\mathbf{Ay} \geq \mathbf{b}$ contains all twelve partition inequalities as shown below for a 3-node complete directed graph.

$$(\mathcal{R}) \qquad \begin{array}{lll}
y_{12} + y_{23} + y_{13} & \geq \lceil d_{12} + d_{23} + d_{13} \rceil & \cdots (r1) \\
y_{12} + y_{13} & \geq \lceil d_{12} + d_{13} \rceil & \cdots (r2) \\
y_{12} + y_{32} + y_{13} & \geq \lceil d_{12} + d_{32} + d_{13} \rceil & \cdots (r3) \\
y_{12} + y_{32} & \geq \lceil d_{12} + d_{32} \rceil & \cdots (r4) \\
y_{12} + y_{32} + y_{31} & \geq \lceil d_{12} + d_{32} + d_{31} \rceil & \cdots (r5) \\
y_{32} + y_{31} & \geq \lceil d_{32} + d_{31} \rceil & \cdots (r6) \\
y_{21} + y_{32} + y_{31} & \geq \lceil d_{21} + d_{32} + d_{31} \rceil & \cdots (r7) \\
y_{21} + y_{31} & \geq \lceil d_{21} + d_{31} \rceil & \cdots (r8) \\
y_{21} + y_{23} + y_{31} & \geq \lceil d_{21} + d_{23} + d_{31} \rceil & \cdots (r9) \\
y_{21} + y_{23} & \geq \lceil d_{21} + d_{23} \rceil & \cdots (r10) \\
y_{21} + y_{23} + y_{13} & \geq \lceil d_{21} + d_{23} + d_{13} \rceil & \cdots (r11) \\
y_{23} + y_{13} & \geq \lceil d_{23} + d_{13} \rceil & \cdots (r12)
\end{array}$$

Claim 3.1 *For a 3-node complete graph, if a vector $y^* \geq 0$ satisfies all partition inequalities (16) with $w^*_a \in \{0, 1\}$ (i.e. y is feasible to R), then $y^* \in LP(C)$, i.e., given the capacities y^*, one can obtain a feasible flow for all commodities.*

Proof: First note that there are 6 distinct partition inequalities with $l(\pi) = 2$ (K2 in short) and 6 different partition inequalities with $l(\pi) = 3$ (K3 in short) with $w^*_a \in \{0, 1\}$. Let $\bar{y}$ be any vector in $\mathcal{R}$. We give a procedure by which a feasible flow can be constructed given the capacity vector $\bar{y}$. For every ordered pair of nodes $(i, j), i \neq j$, we distinguish between the following cases:

- If $\bar{y}_{i,j} \geq d_{i,j}$ then $d_{i,j}$ is sent directly from node i to node j.
- If $\bar{y}_{i,j} < d_{i,j}$ then $\bar{y}_{i,j}$ flow is routed on (i, j) and $d_{i,j} - \bar{y}_{i,j}$ via the third node k.

It is easy to check that the resulting flow satisfies the flow-balance equations, and for all $i \neq j$ the total load on arc (i, j) is:

$$L_{ij} = \min\{d_{i,j}, \bar{y}_{i,j}\} + (d_{i,k} - \bar{y}_{i,k})^+ + (d_{k,j} - \bar{y}_{k,j})^+$$

where $(\alpha)^+ = \max\{0, \alpha\}$. To show that $L_{ij} \leq \bar{y}_{ij}$ for all arcs (i, j), we consider the following two cases.

For any arc (i, j), if $d_{i,j} \geq \bar{y}_{i,j}$, then $(d_{i,k} - \bar{y}_{i,k})^+ = (d_{k,j} - \bar{y}_{k,j})^+ = 0$ due to the K2 inequalities related with the arcs leaving node i and the arcs entering node j, and thus $L_{ij} = \bar{y}_{i,j}$.

On the other hand, when $d_{i,j} < \bar{y}_{i,j}$, then the total load on arc (i, j) equals $d_{i,j} + (d_{i,k} - \bar{y}_{i,k})^+ + (d_{k,j} - \bar{y}_{k,j})^+$. When either the second or the third term is zero, this is at most $\bar{y}_{i,j}$ by K2. When they are both positive, $L_{ij} = d_{i,j} + d_{i,k} + d_{k,j} - \bar{y}_{i,k} - \bar{y}_{k,j}$, which is at most $\bar{y}_{i,j}$ since the K3 inequalities imply:

$$\bar{y}_{i,j} + \bar{y}_{i,k} + \bar{y}_{k,j} \geq d_{i,j} + d_{i,k} + d_{k,j}.$$

We also note that we do not need to round the metric inequalities to show the fesibility of $\bar{y}$. □

3.1 Total Capacity Inequalities

In general the polyhedron $\mathcal{R}$ has fractional extreme points. Consider the case where $d_{21} = 0$ with $d_{ij} = 0.5$ for all other $i \neq j$. One can verify that $y_{12} = 1$, $y_{ij} = 0.5$ for all other arcs, is an extreme point of $\mathcal{R}$. To deal with this case this case we define the *total capacity inequality*

$$y_{12} + y_{21} + y_{23} + y_{32} + y_{13} + y_{31} \geq b \tag{17}$$

where b is the minimum total capacity required for a feasible flow. A few possible values of b are given below.

1. $b = \lceil \frac{1}{2}(\lceil d_{12} + d_{13}\rceil + \lceil d_{12} + d_{32}\rceil + \lceil d_{21} + d_{31}\rceil + \lceil d_{31} + d_{32}\rceil + \lceil d_{21} + d_{23}\rceil + \lceil d_{23} + d_{13}\rceil)\rceil$ from $\frac{1}{2}(r2 + r4 + r6 + r8 + r10 + r12)$.

2. $b = \lceil \frac{1}{3}(\lceil d_{12} + d_{23} + d_{13}\rceil + \lceil d_{12} + d_{32} + d_{13}\rceil + \lceil d_{12} + d_{32} + d_{31}\rceil + \lceil d_{21} + d_{32} + d_{31}\rceil + \lceil d_{21} + d_{31}\rceil + \lceil d_{21} + d_{23}\rceil + \lceil d_{23} + d_{13}\rceil)\rceil$ from $\frac{1}{3}(r1 + r3 + r5 + r7 + r8 + r10 + r12)$.

3. $b = \lceil \frac{1}{3}(\lceil d_{12} + d_{23} + d_{13}\rceil + \lceil d_{12} + d_{32} + d_{13}\rceil + \lceil d_{12} + d_{32}\rceil + \lceil d_{32} + d_{31}\rceil + 2\lceil d_{21} + d_{31}\rceil + \lceil d_{21} + d_{23}\rceil + \lceil d_{23} + d_{13}\rceil)\rceil$ from $\frac{1}{3}(r1 + r3 + r4 + r6 + 2r8 + r10 + r12)$.

Note that for the example given above, $b = 4$ and the corresponding total capacity inequality is violated by the fractional extreme point. Observe that the total capacity inequality is not a rounded metric inequality. It is a Chvátal-Gomory cut obtained from partition inequalities.

3.2 Complete Description of P(C)

We next show that partition inequalities ($\mathcal{R}$) and the total capacity inequality (17) completely define $P(C)$ for the complete graph on 3 nodes. (See [3] for a proof of the following result.)

Proposition 3.2 *For any complete graph with $|V| = 3$, the partition inequalities along with the total capacity inequality completely define $P(C)$.*

Remark: The total capacity inequality can be extended to a valid inequality for graphs with more than 3 nodes if the nodes are partitioned into three sets, V_1, V_2, V_3. The *extended total capacity inequality* is

$$\sum_{i \in V_p,\, j \in V_q}^{p \neq q} y_{ij} \geq b_1, \quad p, q = 1, 2, 3$$

where b_1 is determined similar to that in the total capacity inequality by substituting d_{pq}, $p, q = 1, 2, 3$ with $\sum_{i \in V_p, j \in V_q} d_{ij}$, $p, q = 1, 2, 3$.

4 The Capacity Formulation

The basic solution approach we use is branch-and-cut to obtain the lower bound and the heuristic described below to obtain the upper bound. For details on the branch-and-cut approach, see [6], [7] or [14]. The cuts used in our cutting plane procedure are the partition inequalities, total capacity inequalities, and rounded metric inequalities.

4.1 Initial Formulation

To begin with, we first formulate the problem with only degree inequalities. These inequalities are special cases of the partition inequalities.

$$
(CF) \quad
\begin{array}{lll}
\text{Min} & \sum_{(i,j)\in A} c_{ij} y_{ij} & \\
\text{st.} & y(\delta^+(\{i\})) \geq \lceil \sum_{j\in V-\{i\}} d_{ij} \rceil & \forall i \in S \qquad (18) \\
 & y(\delta^-(\{i\})) \geq \lceil \sum_{j\in V-\{i\}} d_{ji} \rceil & \forall i \in D \qquad (19) \\
 & y_{ij} \in N^+ & \forall (i,j) \in A.
\end{array}
$$

Here S is the set of source nodes and D is the set of destination nodes.

The LP relaxation of formulation (CF) is solved and other valid inequalities (partition, metric, and total capacity), violated by the current solution, are added to the formulation. This procedure of adding new violated inequalities and solving the resulting LP-relaxation is repeated until no more violated inequalities are found. Since the number of constraints generated could be large, inactive constraints are deleted at the end of each iteration. The upper bound heuristic described in Section 4.3 is run at the root node of the branch-and-cut tree to get an initial upper bound. It is then run once every 15 branching nodes, seeking to improve the upper bound.

4.2 Separation Heuristics

In this section we discuss the separation problem for the rounded metric inequalities (15), the partition inequalities (16) and the total capacity inequalities (17). Since separation is hard in general, heuristics have been designed in each case. Even though partition inequalities are a sub-class of rounded metric inequalities, we differentiate between the two because the separation heuristics for the partition inequalities are direct while that for the inequalities (16) require the solution of problem (III) in (13).

Heuristics for partition inequalities

Below we list the separation heuristics for the partition inequality (16) that were computationally most effective.

n-Cut heuristic: This is used to identify partition inequalities with two base sets, i.e. $l(\pi) = 2$. Initially we check connectivity between each source and destination node using only the arcs with $y_{ij} > 0$. If every pair of source and destination nodes is connected, we randomly pick n nodes to form one base set $\bar{V}$ and check whether

$$\sum_{i \in \bar{V}, j \notin \bar{V}} y_{ij} \geq \lceil \sum_{i \in \bar{V}, j \notin \bar{V}} d_{ij} \rceil.$$

If the above inequality is not satisfied, we have a violated inequality. We repeat the above check replacing $\bar{V}$ by $V \setminus \bar{V}$.

We also note that, without rounding, seperating partition inequalities for $l(\pi) = 2$ reduces to the max-cut problem (see [1]). Therefore, it is also possible to form the base sets by using heuristics for the max-cut problem.

n-Partition heuristic: This heuristic is used to identify partition inequalities with n base sets, i.e. $l(\pi) = n, n \geq 2$. In the description of the heuristic we assume that $\alpha(j) = j$. In the first step, n nodes are selected randomly and assigned one each to the n base sets, $V_1, V_2, \cdots, V_n$, so as to minimize the difference between the left hand side and the right hand side of the partition inequality defined on the subgraph induced by the selected nodes. For small values of n this can be done quickly. In the second phase, the remaining nodes are assigned, one at a time. Assume that the nodes $\bar{V}$ have already been assigned to base sets. Let i be the selected node. We restrict attention to the graph G_i induced by the nodes $\bar{V} \cup \{i\}$. The node i can potentially be assigned to any of n base sets keeping the assignment of the nodes in $\bar{V}$ fixed. Let $\text{val}(i, j)$ be the difference between the left hand side and the right hand side of the corresponding partition inequality if node i is assigned to the base set j. Node i is assigned to the base set that minimizes $\text{val}(i, j)$ for $1 \leq j \leq n$. When all nodes are assigned, if the left hand side is less than the right hand side, we have a violated n-Partition inequality.

L-Partition heuristic: Given node lists $L_1 = \{u_i, i = 1, ..., \mid L_1 \mid\}$ and $L_2 = \{v_i, i = 1, ..., \mid L_2 \mid\}$, define $\bar{V} = V \setminus \{L_1 \cup Ł_2\}$. Let $l_i = \mid L_i \mid$. Define the partition $\pi(L_1, L_2) = (V_i, i = 1, ..., l_1 + l_2 + 1)$ where $V_i = \{u_i\}, i = 1, ..., l_1$, $V_{l_1+1} = \bar{V}$, $V_i = \{v_{i-(l_1+1)}\}, i = l_1 + 2, ..., l_1 + l_2 + 1$. Let $\text{diff}(L_1, L_2)$ be the difference between the lefthand side and right hand side of the partition inequality defined by the partition $\pi(L_1, L_2)$.

Given a list L and a node u, let $\{u, L\}$ be the list obtained by placing u at the beginning of L and $\{L, u\}$ be the list obtained by placing u at the end of L. The heuristic proceeds as follows.

1. Set $L_i = \phi, i = 1, 2$ and $j = 1$.
2. Identify min diff$\{(\{L_1, u\}, L_2), \text{diff}(L_1, \{u, L_2\}) \mid u \in \bar{V}\}$. Let u^* be the node defining this minimum.
3. If diff$(\{L_1, u^*\}, L_2) \leq$ diff$(L_1, \{u^*, L_2\})$, set $L_1 = \{L_1, u^*\}$, else set $L_2 = \{u^*, L_2\}$.
4. If diff$(L_1, L_2) < 0$, we have a violated partition inequality. Else set $l = l + 1$. If $l \leq \mid V \mid$, go to 2, else exit.

Both n-Cut and n-Partition heuristics use a random selection process which allow us to run the algorithm several times on the same capacity vector $\mathbf{y}^*$ to obtain several violated inequalities. The L-Partition heuristic, however, identifies at most one violated inequality from a capacity vector. Computationally, the n-Cut and the n-Partition heuristic were very effective in generating violated partition inequalities.

Heuristic for rounded metric inequalities

As a first step we solve problem (III) in (14). In order to avoid an unbounded solution to problem (III), we bound each component of the vector $\mathbf{w}$ from above by 1. A violated metric inequality (15) can thus be identified in polynomial time. Let $\mathbf{w}^*\mathbf{y} \geq \mathbf{dz}^*$ be the inequality (15) obtained. To try and obtain a violated rounded metric inequality (16) we proceed as follows. Let ω be the smallest positive element of the vector $\mathbf{w}^*$. First divide the inequality (15) by ω. If all coefficients in the vector $\mathbf{w}^*/\omega$ are integer and $\lceil \mathbf{dz}^*/\omega \rceil > \mathbf{dz}^*/\omega$, the inequality $(\mathbf{w}^*/\omega)\mathbf{y} \geq \lceil \mathbf{dz}^*/\omega \rceil$ is a violated rounded metric inequality (16) that is stronger than the corresponding metric inequality (15). Thus the corresponding rounded metric inequality is added. If any coefficient in the vector $\mathbf{w}^*/\omega$ is not integer or $\lceil \mathbf{dz}^*/\omega \rceil \leq \mathbf{dz}^*/\omega$ we simply add the violated metric inequality (15). This procedure was very effective in practice.

Heuristic for total capacity inequalities: Each time we run the n-Partition heuristic described above, for $n = 3$, we simultaneously check if the corresponding total capacity inequality (17) is violated. In case both the partition and total capacity inequality is violated, only the total capacity inequality is added.

To give an idea about the relative importance of the valid inequalities, we next present the details of the cuts generated for some of the problems we studied. In Table 1, **n-Cut** is the percentage of cuts generated by the n-Cut heuristic. **3-Part** is the percentage of cuts generated by the n-Partition heuristic for $n \leq 3$. **+4-part** is the percentage of cuts generated by the n-Partition heuristic for $n \geq 4$. **TCI** is the percentage of total capacity inequalities (18) generated. **RMI** is the percentage of rounded metric inequalities (16) generated. **Total** is the total number of cuts generated. **Round** is the percentage of metric inequalities (15) generated that are rounded using the heuristic. **Tight** is the number of tight cuts at the root node.

Prob.	n-Cut	3-Part	+4-part	TCI	RMI	Total	Round	Tight
MS1	7.15	19.2	55.1	2.25	16.21	22951	99.90	93
MS2	6.75	19.8	50.1	2.70	20.40	18306	99.50	88
MS3	9.60	19.3	55.1	2.50	13.30	19702	99.80	95
MS4	8.40	19.7	54.7	2.90	14.00	24490	99.60	100
MS5	9.50	21.0	50.5	2.60	16.00	20479	99.80	92
MS6	5.70	16.5	61.0	2.30	14.30	25309	98.90	116

Table 1: Cuts generated for problems in Table 6

Note that a majority of the cuts are generated by the n-partition heuristic (over 50 % of the cuts come from partitions of size 4 or more). Also over 99 % of the violated metric cuts are rounded to obtain stronger cuts. A key observation is that the number of tight cuts is very small. Thus each LP is quick to solve since only the tight cuts are retained from one iteration to the next. However note the large number of cuts generated by our heuristics. One key to speeding up the solution process may be to better target the violated cuts added.

4.3 Generating Upper Bounds

The heuristic we propose utilizes the LP solution and the polyhedral cutting-plane technique to reach a feasible solution serving as an upper bound. It consists of two major steps that are repeated iteratively. They are called *variable fixing* and *new cut addition*.
Variable fixing: Assume we have a fractional solution y^*. For all variables y_a^* that are integral, we fix lower bounds to be $y_a \geq y_a^*$. Among the fractional variables, select the variable with the smallest value of $c_a(\lceil y_a^* \rceil - y_a^*)$. The lower bound for this variable is fixed to be $\lceil y_a^* \rceil$. Go to the new cut addition step.
New cut addition: The modified LP-relaxation is solved and new cutting planes (partition, metric and total capacity inequalities violated by the current solution) are added to the current formulation. The cycle of adding new cuts and solving the resulting LP-relaxation is repeated, until no more violated inequalities can be identified. At this stage, if we have a fractional solution, the variable fixing step is repeated. If the solution is integer and satisfies all metric inequalities (this check can be made in polynomial time) we stop. The resulting integer capacity vector admits a feasible flow.

Note that each time we go through the variable fixing step, at least one additional variable gets an integer lower bound. Thus this step can be repeated only a limited number of times. The heuristic is fairly time consuming. However the quality of the solution obtained is good.

5 The Multicommodity Flow Formulation

The approach we follow to solve the problem via the Multicommodity Flow formulation borrows heavily from that used in [4]. The problem studied there differs from MCCI in that (a) there are two "modularities" of capacities that can be added to edges: that is, there are two numbers B and λ, such that we add capacity to edges in integral multiples of B and of λB; (b) we pay for flow costs; and (c) for every edge (i, j) there is a reverse edge (j, i) and both edges must have the same capacity (this side constraint arises from certain real-life capacity allocation issues). However, to some degree the problems are similar enough that we were able to adapt the approach of [4] for use here.

The basic strategy is as follows: We use the source-aggregated multicommodity formulation and go through a standard cutting-plane phase based on facet-defining inequalities described below. We have decided to include the degree inequalities (18) and (19) in the initial formulation, most of which were otherwise violated and added to the formulation early on. After the cutting-plane phase, we use the resulting extended formulation and the fractional optimal solution to obtain upper bounds. The final step is applying branch-and-bound.

5.1 Cutting Planes

For this study we used three classes of facet-defining inequalities. For U, W disjoint subsets of vertices, we denote by $d(U, W)$, the sum of demands with source in U and destination in W. The first inequality is the basic cut inequality.

Cut inequality. Let U be a subset of vertices. The cut inequality corresponding to U is,

$$y(\delta^+(U)) \geq \lceil d(U, V - U) \rceil \tag{20}$$

whose validity is clear. See [1], [8] for similar inequalities. It is possible to show that these inequalities are facet-defining when U and its complement are strongly connected and $d^+(U)$ is not integral.

In spite of their apparent simplicity, the cut inequalities are very effective and usually violated in large amounts. However, they are not quite enough for solving the problem, and we used two additional families of strong inequalities. Recall that x_a^k is the flow of commodity k on the directed edge a.

Flow-cutset inequality. Let U be a subset of vertices, J a subset of commodities with source in U (i.e., a subset of U). Let A_1, A_2 be a partition of $\delta^+(U)$. We define $r(J, V-U) = d(J, V-U) - \lfloor d(J, V-U) \rfloor$ if $d(J, V-U)$ is not integral and $r(J, V-U) = 1$ otherwise. Then

$$\sum_{k \in J} \sum_{a \in A_1} x_a^k + r(J, V-U) \sum_{a \in A_2} y_a \geq r(J, V-U) \lceil d(J, V-U) \rceil \tag{21}$$

is a valid inequality. It can be derived from the original formulation as a mixed-integer rounding inequality, see [9]. The last type of inequality is the 3-partition inequality we have already seen above (also see [8] for a similar inequality).

3-partition inequality. Let U_1, U_2, U_3 be a partition of the vertex-set into three classes. Then, the following partition inequality

$$y(\delta(U_1, U_2)) + y(\delta(U_1, U_3)) + y(\delta(U_2, U_3)) \geq \lceil d(U_1, U_2) + d(U_1, U_3) + d(U_2, U_3) \rceil \tag{22}$$

is valid.

We also note that inequalities (21) and (22) are facet-defining under approppriate conditions.

5.2 Cutting Strategy

Our strategy for finding violated facet inequalities is subset-based. Using two classes of heuristics for separating the cutset inequalities, we generate and maintain a list of "interesting" subsets of vertices as indicated below. Essentially, a subset is interesting if the corresponding cut inequality is tight or violated. Using further heuristics, we look at flow-cutset and 3-partition inequalities *involving interesting subsets only* to look for violations. The intuition here is that if a cut inequality is currently violated or fairly tight, some flow-cutset or 3-partition inequality involving U may be violated, either right now or in the next iteration of the cutting-plane procedure. To augment the list of interesting subsets, we occasionally "kick" them, that is to say, given any interesting subset S, we also check to see if S minus a singleton, or S plus a singleton, should also be considered interesting.

Thus the main part of our algorithm is a procedure for detecting violated cut inequalities. Given a fractional point, feasible for the multicommodity formulation, we can show that the problem of detecting a violated cut inequality is NP-hard (Barahona [2] also proved a related result). We use two heuristics for this purpose, a "fast" heuristic and an "extensive" heuristic.

Fast heuristic. In this heuristic we essentially contract any edge (i, j) that is currently very slack for the capacity constraint (3). In the resulting graph, we choose subsets of nodes that are "close" to each other and check the cut inequality for violations. More precisely, we use the following **Subset Generation Heuristic :**

- Given a fractional solution, let

$$s_{i,j} = y_{ij} - \sum_k x_{ij}^k. \tag{23}$$

- Fix parameters $\alpha \in Z_0^+$, $\beta \in R^+$ and define, for $(i, j) \in A$

$$\bar{w}_{i,j} = \begin{cases} 1 & s_{i,j} < \beta \\ 0 & s_{i,j} \geq \beta \end{cases} \tag{24}$$

- Find $dist(i, j)$ = shortest path lengths using $\bar{w}$.

$$\begin{aligned} S^{out}(i) &= \{j \ : \ dist(i, j) \leq \alpha\} \\ S^{in}(i) &= \{j \ : \ dist(j, i) \leq \alpha\} \end{aligned}$$

- Check the sets $S^{out}(i)$ and $S^{in}(i)$ for violations, and "kick" them.

For β we used values $0.05, 0.10, 0.15$ and 0.20, and we set $0 \leq \alpha \leq 2$. The idea behind this heuristic is that if we find a cut each of whose edges is tight, then most likely the corresponding cut inequality should be at least tight, if not violated.

This heuristic runs quickly, and for some problems it was enough to obtain the optimal solution. For other problems it failed to find some of the "crucial" subsets, and here the extensive heuristic became effective.

Extensive heuristic.. This heuristic is used when the fast heuristic is struggling (e.g., no further inequalities are being separated and the solution is far from integral). Say that a subset is *critical* if it and its complement are strongly connected. The extensive heuristic maintains two lists of critical subsets, a "master" list and an "active" list. The active list is a subset of the master list and is usually much smaller.

In the typical iteration of the cutting plane algorithm, we check the current fractional point against all cut inequalities in the active list to look for violations. Inequalities that are active but not violated are kept active if they are tight enough (say, within 5 % of being tight) and are otherwise removed. From time to time, the algorithm scans the master list in random order, and promotes inequalities to active according to the same criterion, until the entire master list has been scanned or “enough” inequalities have been promoted. The initial active list is obtained in this way. Finally, we use the “kick” procedure as indicated above.

The advantage of using this heuristic is that if the active list is rich enough at the start of the algorithm then the extensive heuristic will detect many violations immediately, thus improving the quality of the formulation early on. This has certainly been our experience with the problems we handled. The only question concerns how to generate the master list, and how to maintain it. The algorithm should occasionally move inequalities into and out of the master list.

The crucial element here is that while real-life networks can be moderately large, say, up to 90 nodes, when they are this large they are extremely sparse, almost tree-like (see [12]). A more typical scenario involves a network with no more than 30 nodes which is two-connected but still somewhat sparse. In such networks the number of critical subsets is fairly small. In fact, using an appropriate algorithm, for all the real-life networks we have been able to *enumerate* all the critical sets in just a few minutes of computation. Given that it may require hours to solve the problems, this is not an onerous step. Thus, in the problems we handled the master list was in fact the list of all cut inequalities, stored and handled efficiently as indicated above. In the largest example the master list contained roughly $62,000$ inequalities, and typically only a few hundred were active.

Which is more efficient, the quick heuristic or the extensive heuristic? In most of the more difficult problems the extensive heuristic performed better, but not always. It is a heuristic because the active list has to be selected carefully, and more work needs to be done here. Some of these issues will be examined in a future paper.

As indicated above, we separate the flow-cutset and 3-partition inequalities by restricting separation to carefully chosen “interesting” subsets. In the current implementation, all the active sets are declared interesting (and we apply the “kick” procedure). This appears to have been adequate, but additional work is needed. In any case, the intuition is that if our list of interesting subsets is indeed nontrivial, then we should be able to detect significant violations. To check for violations of a flow-cutset inequality for a given subset U, we look at subsets J of U of small cardinality, say $|J| \leq 3$ or of large cardinality ($|U - J| \leq 3$). We expect the general case to be NP-hard. Concerning the 3-partition inequalities, we do the following: We look at three-partitions consisting of (a) two singletons and an interesting subset; or (b) a singleton, an interesting subset, and the rest of the vertices; or (c) we choose two interesting subsets $U_1, U_2 \subset V$ such that $U_1 \subset U_2$, and construct a partition using U_1, $U_2 \setminus U_1$ and $V \setminus U_2$.

5.3 Generating Upper Bounds

The formulation, produced by the cutting-plane procedure described above is usually effective in the sense that, the final fractional solution vector contains many zeroes. In general, most of the capacity variables are either integral or close to being integral. We obtain upper bounds by rounding these variables to the nearest integer, and then applying branch-and-bound restricted to the remaining columns. This restricted problem can be solved to near optimality very quickly and the resulting solutions give tight upper bounds for the original problem. In the final version, we decided to round a variable to its nearest integer if it was within 10% of being integral.

In our computational experience, this simple heuristic has proved to be very fast, and we were able to find very tight upper bounds for most of the problems. After running the heuristic we then switch to branch-and-bound on all the variables. As the tables indicate, this strategy has proved effective for the problems at hand.

However, MCCI has a very special structure, and it should be possible to develop a fast combinatorial heuristic. This is a topic for future work, and we hope to report on it in the future.

6 Computational Experiments

In our experiments we ran both formulations on two classes of real-life problems available to us. The first one is based on the problems studied in [5] where the network is for the New York area. In these problems the network has 15 nodes and 44 arcs; and the traffic matrix is fully dense. The second data set is obtained from Stoer [11] and we call them "Norwegian problems". In these problems the network has 27 nodes (corresponding to Norwegian cities) and a total of 102 arcs. In this data set there are less than 20 demand pairs and the capacity costs are the road distances between the cities. These problems are also studied in [8] but the costs there are earth distances rather than road distances. As seen in Tables 2 and 3, both formulations are effective and all four original problems available to us are solved to optimality in a reasonable amount of computation time. We next present a key to these tables.

Capacity Formulation: In Table 2, the size of problems are specified by the number of nodes $|V|$, the number of arcs $|A|$, and number of demand pairs $|K|$. **XLP** is the lower bound obtained at the root node of the branch-and-cut tree. **ALP** is the cpu time to reach the root node before branching. **UBH** is the upper bound on applying the heuristic at the root node. **BLB** is the lower bound obtained at the end of the branch-and-cut procedure. **BUB** is the upper bound obtained at the end of the branch-and-cut procedure. **Total** is the total solution time.

Prob.	$\|V\|$	$\|A\|$	$\|K\|$	XLP	ALP (sec.)	UBH	BLB	BUB	GAP (%)	Total (sec.)
NY1	15	44	210	24802	85	24804	24804	24804	0.00	347
NY2	15	44	210	77602	76	85860	78855	78855	0.00	127
NW1	27	102	18	318.3	157	328.9	323.5	323.5	0.00	203
NW2	27	102	19	641.5	122	724.8	654.7	654.7	0.00	2626

Table 2: Original New York and Norwegian Problems – Capacity Formulation

Multicommodity Formulation: In Table 3 below, **LP** denotes the value of the first linear program (which includes the degree constraints) **XLP** is the value after adding cutting planes, **cut** is the total time spent by the cutting plane algorithm, **UB-H** is the value of the upper bound found by the heuristic, **P-GAP** is the percent gap between the best upper and lower bounds for the restricted problem, **heur** is the time spent by the heuristic, **BLB**, **BUB**, and **bb** are the best lower and upper bound found and time obtained by branch-and-bound on all the columns. **GAP** is the percent gap between the best upper and lower bounds found by the overall algorithm, and **Total** is the overall running time. We used CPLEX, v. 3.0 as the LP optimizer and times are on a SUN Sparc 10/51. We note that Problem NW1 was solved to optimality in the cutting phase and in Problem NW2 the lower bound was essentially obtained after 166 seconds, but the algorithm continued looking for violated inequalities for a total of 1550 seconds.

Prob.	LP	XLP	cut (sec)	UB-H	P-gap (%)	heur (sec)	BLB	BUB	bb (sec)	GAP (%)	Total (sec)
NY1	23941	24794	7	24804	0.04	2	24804	24804	4	0.00	13
NY2	70602	77977	8	79215	1.58	5	78855	78855	52	0.00	65
NW1	203.2	323.5	49	-	-	-	-	-	-	0.00	49
NW2	502.5	649.8	1583	656.7	1.06	2	654.7	654.7	190	0.00	1775

Table 3: Original New York and Norwegian Problems - Multicommodity Formulation

We obtained further problems by using the Norwegian network with the original cost data, and generating the traffic demands randomly. In Tables 4 and 5 we restrict our attention to problems with a single source and multiple destinations (broadcast problems). When generating the demands, we first randomly pick the source node, and then choose a fixed number of demand nodes. To see the effect of changing the number of demand nodes, we generated problems with 10, 15 and 20 demands. In the following tables, **Source** denotes the number of source nodes.

Prob.	$\|K\|$	XLP	ALP (sec.)	UBH	BLB	BUB	GAP (%)	Total (sec.)
SS1	10	292.7	62	313.5	298.3	298.3	0.00	117
SS2	10	626.4	160	626.4	-	-	0.00	160
SS3	15	497.4	91	497.4	-	-	0.00	91
SS4	15	876.6	111	946.6	897.9	897.9	0.00	270
SS5	20	784.8	215	806.3	803.2	803.2	0.00	357
SS6	20	988.9	94	1034.9	1003.1	1003.1	0.00	188

Table 4: Norwegian Network with Single Source – Capacity Formulation

As seen in Tables 4 and 5, both formulations can solve these problems to optimality. The Capacity Formulation takes under 200 seconds on the average, and can solve two of the problems to optimality without any branching. The Multicommodity Formulation on the other hand, solves all of the problems in the cutting phase and takes under 20 seconds on the average.

Prob.	LP	XLP	cut (sec)	heur (sec)	bb (sec)	GAP (%)	Total (sec)
SS1	204.9	298.3	7	-	-	0.00	7
SS2	563.8	626.4	6	-	-	0.00	6
SS3	431.1	497.4	7	-	-	0.00	7
SS4	814.7	897.9	31	-	-	0.00	31
SS5	723.5	803.2	27	-	-	0.00	27
SS6	935.1	1003.1	7	-	-	0.00	7

Table 5: Norwegian Networks with Single Source – Multicommodity Formulation

We next (randomly) generated problems with many origin-destination pairs. In Tables 6 and 7, we consider 67 source destination pairs but alter the number of sources. We note that none of the problems with multiple sources (MS1-MS6) are solved to optimality within the prescribed time limit of 18,000 seconds for the Capacity Formulation, and 3,600 seconds for the Multicommodity Formulation. For the Multicommodity Formulation, the total time limit is divided among the the cutting and bounding steps of the algorithm as follows: 2050 seconds for cutting, 1000 seconds for the heuristic and 550 seconds for pure branch-and-bound.

Prob.	source	$\|K\|$	XLP	ALP (sec.)	UBH	BLB	BUB	GAP (%)	Total (sec.)
MS1	6	67	2918.6	1273	3027.3	2930.3	3027.3	3.21	18000
MS2	6	67	2926.0	1088	3123.4	2937.9	3013.6	2.51	18000
MS3	12	67	2960.5	2657	3093.6	2970.1	2979.4	0.32	18000
MS4	12	67	3219.0	4318	3373.0	3221.7	3309.9	2.66	18000
MS5	18	67	3166.7	5403	3333.4	3169.0	3282.9	3.47	18000
MS6	18	67	2555.1	2216	2633.4	2561.9	2633.4	2.71	18000

Table 6: Norwegian Network with Multiple Sources – Capacity Formulation

As seen in Tables 6 and 7, the quality of the solutions obtained for problems with with multiple sources is fairly good, with an average gap of 2.3 percent for the Capacity Formulation, and 1.0 percent for the Multicommodity Formulation.

Prob.	LP	XLP	UB-H	P-gap (%)	BLB	BUB	GAP (%)	Total (sec)
MS1	2753.4	2925.7	2976.3	0.83	2933.2	2976.3	1.45	3600
MS2	2791.5	2943.1	2978.2	0.01	2955.3	2978.2	0.77	3600
MS3	2829.5	2969.4	2985.3	0.30	2978.9	2978.9	0.00	3600
MS4	3067.9	3225.9	3270.9	0.42	3232.3	3256.8	0.75	3600
MS5	2997.6	3169.6	3246.6	2.32	3174.8	3246.6	2.21	3600
MS6	2391.2	2563.4	2592.4	1.09	2566.9	2592.4	0.98	3600

Table 7: Norwegian Network with Multiple Sources – Multicommodity Formulation

Lastly, we generated problems with fully dense traffic matrices. In these problems (FD1-FD6), the number of source nodes is 27 and $|K| = 27 \times 26 = 702$. As in Problems MS1-MS6, we ran the two algorithms for 18,000 and 3,600 seconds respectively and the related computational results are presented in Tables 8 and 9. As seen in these tables, the Multicommodity Formulation performed better on these problems.

We also note that the run times for the Capacity Formulation (Tables 6 and 8) are somewhat deceptive because fairly good lower bounds are available at the root node of the branch-and-cut tree. On average the lower bound at the root node is within 2.5 percent of the best upper bound (this is only marginally worse than the gap at the end of 18,000 seconds. The average time to reach the root node however is under 5,000 seconds). This indicates the effort spent in the branch-and-cut tree with relatively minor improvements. The time it takes to solve the problem at the root node increases with the number of commodities (most of the increase comes from the time spent separating metric inequalities). However the quality of the solution at the root node stays quite consistent.

Prob.	$\|K\|$	XLP	ALP (sec.)	UBH	BLB	BUB	GAP (%)	Total (sec.)
FD1	702	29522	5805	30782	29549	30032	1.61	18000
FD2	702	29508	5688	30666	29518	30211	2.30	18000
FD3	702	98054	6090	115675	98133	100748	2.60	18000
FD4	702	97853	8827	99839	97873	99839	1.97	18000
FD5	702	58872	6304	60178	58909	60178	2.11	18000
FD6	702	58687	9301	59696	58692	59696	1.68	18000

Table 8: Norwegian network with fully dense demands – Capacity Formulation

Prob.	LP	XLP	UB-H	P-gap (%)	BLB	BUB	GAP (%)	Total (sec)
FD1	29594	29719	29804	0.28	29722	29804	0.28	3600
FD2	29583	29703	29835	0.44	29708	29835	0.43	3600
FD3	98575	98696	98829	0.13	98698	98829	0.13	3600
FD4	98347	98465	98556	0.09	98472	98556	0.08	3600
FD5	59114	59212	59337	0.20	59217	59337	0.20	3600
FD6	58923	59053	59130	0.12	59055	59130	0.13	3600

Table 9: Norwegian network with fully dense demands – Multicommodity Formulation

References

[1] Barahona, F., "Network design using cut inequalities," preprint, IBM, December 1994.

[2] Barahona, F., personal communication.

[3] Bienstock, D., Chopra, S., Günlük, O., and Tsai, C-Y, "Minimum Cost Capacity Installation for Multicommodity Network Flows," *CORE Discussion Paper*, Université Catholique de Louvain, Belgium, (1995).

[4] Bienstock, D. and Günlük, O., "Computational Experience with a Difficult Mixed-Integer Multicommodity Flow Problem," *Math. Programming*, **68**, 213-238, (1995).

[5] Bienstock, D. and Günlük, O., "Capacitated Network Design - Polyhedral structure and Computation," (May 1994) to appear, *ORSA Journal on Computing.*

[6] Hoffman, K. and Padberg, M., "Improving LP-Representation of Zero-One Linear Programs for Branch-and-cut", *ORSA Journal on Computing*, **3**(2), pp.121-134, (1991).

[7] Hoffman, K. and Padberg, M., "Solving Airline Crew Scheduling Problems by Branch-and-cut", *Management Science*, **39**(6), pp.657-682, (1993).

[8] Magnanti, T. and Mirchandani, P. and Vachani, R., "Modeling and Solving the Two Facility Capacitated Network Loading Problem", *Operations Research*, to appear.

[9] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York (1988).

[10] Pulleyblank, W.R., "Polyhedral Combinatorics", in Handbooks in OR & MS, Vol. 1, G.L. Nemhauser at al., eds., North-Holland, Amsterdam, 1989, pp. 371-446.

[11] Stoer, M., private communication.

[12] Stoer, M and Dahl, G., "A Polyhedral Approach to Multicommodity Survivable Network Design", *technical report*, University of Oslo, Department of Informatics, (1995).

[13] Onaga, K and Kakusho, O., "On Feasibility Conditions of Multicommodity Flows in Networks", *IEEE Transactions on Circuit Theory*, CT-18, No. 4, pp. 425-429.

[14] Padberg, M. and Rinaldi, G., "A Branch-and-Cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems", *SIAM Review*, **33**, pp.60-100, (1991).

[15] R.L. Rardin and L.A. Wolsey, "Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network network flow problems",*European Journal of Operations Research*, (November 1993).