Mathematical Programming 68 (1995) 213–237

Computational experience with a difficult mixed-integer multicommodity flow problem☆

D. Bienstock*, O. Günlük

Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027, USA

Received 27 April 1993; revised manuscript received 17 June 1994

Abstract

The following problem arises in the study of lightwave networks. Given a demand matrix containing amounts to be routed between corresponding nodes, we wish to design a network with certain topological features, and in this network, route all the demands, so that the maximum load (total flow) on any edge is minimized. As we show, even small instances of this combined design/routing problem are extremely intractable. We describe computational experience with a cutting plane algorithm for this problem.

Keywords: Mixed-integer programming; Multicommodity flows

1. Introduction

1.1. Problem definition

We are interested in the following optimization problem. Given an $n \times n$ matrix T (whose i, j entry is indicated by t_{ij}), and an integer $p > 0$,

(1) construct a simple directed graph D with node set $1, \ldots, n$, where each node has indegree and outdegree equal to p, and

(2) in D, simultaneously route t_{ij} units of flow from i to j, for all $1 \leqslant i \neq j \leqslant n$,

so as to minimize the maximum aggregate flow on any edge of D.

This problem is briefly motivated as follows. An important problem in communication networks is to route existing traffic requests so as to keep congestion levels as low as

☆ This research was partially supported by a Presidential Young Investigator Award and the Center for Telecommunications Research, Columbia University.

* Corresponding author.

SSDI 0025-5610(94)00050-4

possible. One way to approach this problem is to route so that the maximum total flow on any edge is as small as possible. This leads to the mathematical problem known as the *maximum concurrent flow* problem, which has received extensive attention (see [16], [5], [10] and [11] for some computational experiments).

In the operation of lightwave networks it is possible to alter the network topology, within certain limitations. This feature can be used to handle changing traffic patterns. We refer the reader to [9], but in essence the situation is as follows. Each node in the network is equipped with a small number of tunable transmitters and receivers. If a certain node, u, tunes a transmitter to the same frequency that another node v has tuned a receiver, then flow can be sent directly from u to v. It is assumed that no frequency can be shared by more than two nodes at a time. Consequently, the set of ordered node pairs (u, v) corresponding to the frequencies in use yields a directed graph (the *logical network*) which can then be used to route traffic requests. It is further assumed that traffic can be routed in a *divergent* manner, i.e., if a certain request specifies that a given amount of traffic is to be sent from a node u to a node v, then it is permitted to use several simultaneous paths, each carrying some fraction of the total desired amount.

In this framework, the ideal way to deal with the congestion problem seemingly would be to view demands as constantly changing, and to change network structure in 'real time' to adapt to new conditions. However, in practice one would not wish to frequently rearrange the existing network, since it would be very disruptive and expensive to continuously reroute existing traffic. In fact, the network should probably not be rearranged more frequently than once every few hours.

In this manner one arrives at the abstract problem described above.

(Remarks: (1) As indicated, the graph D is assumed to be simple (i.e., no parallel edges). If parallel edges are allowed the problem is similar and the cutting planes we will describe later remain valid, but we will not consider this generalization in this paper. (2) In a more general version of the problem, for each node we have specified upper and lower bounds on the indegree and outdegree. This appears to be a much more difficult problem.)

We note here that, given a fixed network D, the task of routing the commodities to minimize the maximum load is in fact a linear program (this is a special case of the *maximum concurrent flow* problem) and can therefore be efficiently solved. Our problem involves routing and also choosing the network, and is substantially more difficult (it is NP-hard even for $p = 1$). In practice, one would wish to have a relatively fast heuristic that generates good solutions. We have observed that even small instances can be extremely intractable (see below), so that several hours of running time on a powerful computer may in fact be necessary to get good solutions for a large instance.

A heuristic approach for this problem is given in [9]. The heuristics generate both solutions and lower bounds (for the min–max load) and are fast. However, as reported in [9] the bounds produced by these heuristics when applied to some small instances (with fully dense demand matrices) were usually rather far apart, with typical gaps between lower and upper bounds of the order of 20% to 30%.

In this paper, we report on computational experiments with a cutting plane algorithm for a mixed-integer programming formulation of the problem, which is used to obtain good

lower bounds. Our experiments are all for the case $p=2$, motivated by the study in [9]. The cutting plane algorithm yields good lower bounds and also an extended formulation that appears effective as a starting point for branch-and-bound.

There are some noteworthy features about the problem:

(i) The mixed-integer program is extremely difficult, with very large gaps between the continuous relaxations and the integral optima,

(ii) the linear programs to be solved appear to be quite hard, and

(iii) in a practical setting, the time available for computation would be limited, of the order of a few hours, say.

Our early computational experience is encouraging: the gaps in the 'bench-mark' problems in [8] were substantially reduced in most cases (and never worsened), within a few minutes of computation. We also report on similar results for much larger, less than fully dense, randomly generated problems.

1.2. Mixed-integer programming formulation

We will work with the following mixed-integer programming formulation of the problem (notation as in 1.1). For each ordered pair of nodes i, j there is a $\{0, 1\}$-variable x_{ij}, set to 1 if the edge (i, j) is put in the network, and set to 0 otherwise. For each commodity k and ordered pair i, j, there is a continuous variable f_{kij}, that measures the quantity of flow of commodity k on the edge (i, j) (more below on what constitutes a commodity). For a commodity k and a node i, we denote by s_i^k the *net demand* of commodity k at i. The overall formulation is:

$$\min \quad z$$

$$\text{s.t.} \quad \sum_{j \neq i} x_{ij} = p, \quad \text{for all } i, \tag{1}$$

$$\sum_{j \neq i} x_{ji} = p, \quad \text{for all } i, \tag{2}$$

$$f_{kij} \leqslant M_{ij}^k x_{ij} \quad \text{for all } k,\ i \neq j, \tag{3}$$

$$\sum_{j \neq i} f_{kji} - \sum_{j \neq i} f_{kij} = s_i^k \quad \text{for all } i, k, \tag{4}$$

$$\sum_{k} f_{kij} \leqslant z \quad \text{for all } i \neq j, \tag{5}$$

$$0 \leqslant f_{kij}, \quad x_{ij} \in \{0, 1\}, \quad \text{all } k, \quad \text{all } i \neq j.$$

Equations (1) and (2) are degree constraints. Equation (3) indicates that we can route flow on edge (i, j) only if the edge is there (M_{ij}^k is an appropriately large quantity – more on this later). Equation (4) is a flow conservation equation, and equation (5) measures the load on edge (i, j).

We will denote this mixed-integer program by ICONG(p). Except for variable z, the constraints involving it and the objective function, this can be regarded as a (multicom-

Table 1

Formulation	problem **quasiunif1**			problem **quasiunif2**			problem **ring**		
	Value	Time	Quality (%)	Value	Time	Quality (%)	Value	Time	Quality (%)
agg	9.92	4.47	16.04	11.72	4.98	17.89	33.81	6.15	27.27
disagg	15.82	286.78	25.58	17.31	268.25	26.42	38.19	297.83	30.80
agg + cuts	57.78	16.82	93.45	59.64	14.67	91.05	105.43	21.13	85.02
disagg + cuts	57.96	216.30	93.74	60.22	188.50	91.94	115.98	286.15	93.53

modity) uncapacitated fixed charge network flow problem, see [13]. There are two important points concerning the formulation that we would like to bring up here.

(a) We may choose commodities to be either disaggregate (i.e., every nonzero entry in the demand matrix yields a distinct commodity – in other words, commodities correspond to source–destination pairs) or aggregate. For example, we may view all demands with the same source as constituting a commodity ([15] uses the terminology 'multicommodity' to refer to the disaggregate version, but here we will not because the overall problem is already multicommodity). The 'fine grain' disaggregate formulation for general (uncapacitated) fixed-charge network flow problems is stronger, and it may also be possible to use stronger valid inequalities than for the aggregate version. On the other hand, and in particular in the case of ICONG(p) the linear programs arising in the disaggregate version will be *extremely large* and computationally expensive. A system of inequalities, called 'dicut collection inequalities', that yields the projection of the disaggregated formulation on the space of the aggregated formulation has been found [15]. But the separation problem for dicut collection inequalities appears difficult. In Table 1 we provide information concerning the aggregated and disaggregated formulations for three problem instances of ICONG(2) considered in [9]. For each problem, we list data for four formulations: (1) aggregated, (2) disaggregated, (3) aggregated with some cuts added, and (4) disaggregated with the same cuts as in (3) added.

The column labeled 'Quality' lists the ratio of the LP value to the best upper bound known for the problem (which are, respectively, within 5.5% of optimality for problem **quasiunif1**, within 3% of optimality for problem **quasiunif2** and optimal for problem **ring**). Times are in seconds on a Sun Sparc2, using CPLEX 2.0 with primal steepest edge pivoting (consistently the best choice). The results above are typical. As expected, the disaggregated formulation is stronger than the aggregated one, but not spectacularly so (even with cuts added). In particular, the disaggregated formulation, by itself, does not cut the gap to the optimum to a few percent (in contrast to the more standard instances of FCNF as reported in [15] and [1]). On the other hand, the disaggregated formulation is substantially more expensive computationally: here we note that the three problems listed above have $n=8$ (so the aggregated formulation has 449 columns and the disaggregated one, 1233, both after eliminating redundant columns). These results are typical. We would expect the disaggregated formulation to be prohibitively expensive for larger problems, and yet not a substantial improvement on the aggregated formulation. Consequently, we focused our work on the aggregated formulation, and all results reported below are for it.

(b) The choice of the constants M_{ij}^k in the variable upper-bound inequalities (3) above is important – in our experiments the quality of the formulation was highly dependent on keeping these numbers as small as possible. The following is a possible choice. For a given node (i.e., commodity) k, and edge (i, j) we set $M_{ij}^k = \sum_{j \neq k} t(k, j)$. Below we will discuss how to strengthen this bound. We also note that variable upper-bound inequalities make a linear program degenerate and so we might expect the LP-relaxation of ICONG(2) to be a difficult linear program, and it is, but not just for this reason. At first glance, replacing all the inequalities (3) corresponding to a single edge (i, j) with one inequality of the form $\sum_k f_{kij} \leqslant U x_{ij}$ might appear to be a good strategy. But in our experiments, particularly with larger problems, this made the formulation significantly weaker. On the other hand, the number of inequalities (3) is very large and typically a small number of them are active. We used these inequalities as cutting planes.

1.3. How difficult is ICONG(2)?

Even though the gap between the LP relaxation of ICONG(2) and the optimum tends to be very large, it is conceivable that the problem could be solvable using branch-and-bound (possibly after adding some cutting planes) especially in the case of small instances. Early in our research, we made available to several groups of leading researchers the eight-node problem instance labeled **quasiunif2** in Table 1 (56 0–1 variables). (Remark: the number of digraphs on eight nodes, with indegrees and outdegrees equal to 2, exceeds 10^9.) In fact, we made two formulations available. The first one is the standard one as given above, and its LP-relaxation value is 11.72. The second one contains a number of additional valid inequalities, that raise the LP-relaxation value to 59.06. We will refer to these two formulations as the *weak* and the *strong* formulation, respectively. The best lower and upper bound for this problem are 63.99 and 65.67, both obtained by W. Cook [4] by running his branch-and-bound code on our extended formulation as described in later sections.

Several experimental and commercially available codes were run on both instances. When run on the weak formulation, none of the codes obtained a lower bound higher than 30.00, and when run on the strong formulation, none of the codes improved the LP lower bound, in both cases despite very substantial running times (several days, using large machines) and branch-and-bound trees with hundreds of thousands of nodes.

On the positive side, all of the codes found integral solutions that (*a posteriori*) are within 2 or 3% of the likely optimum. In particular, using a variation of the strong formulation (using some of the inequalities given below) L. Wolsey very rapidly found a solution of cost 66.20: he did this by fixing the 0–1 variables set to 1 by the linear program (6, out of 16 that will equal 1 in any feasible solution) and *exhaustively* running branch-and-bound on that branch of the tree (the lower bound did not improve, however). The final note in this story is that when the strong formulation was augmented to include many more of our inequalities, several codes significantly tightened the bounds and reduced the computational overhead ([4], [2]).

It is therefore clear that formulation ICONG(2) must be strengthened. Further, branch-and-bound may not be a practicable option when dealing with larger problems. This is due

to the fact that the linear programs are very difficult. For example, in several instances with $n=20$, the solution of each linear program required on the order of 5 minutes when the variable upper bounds were omitted, and this grew very quickly once cutting planes were added to the formulation. As stated before, in the 'real-life' setting only a few hours may be available for dealing with a problem. So we would seek an algorithm that yields good bounds while solving a limited number of linear programs.

As a simple example of why these problems are combinatorially difficult, suppose that the demand matrix is given by:

$$t_{ij} = \begin{cases} 1, & \text{if } j=i+1 \text{ and } i<n \text{ or } i=n \text{ and } j=1, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where we assume $n \geqslant 3$. With $p=2$, it can be shown that the value of the problem is $\frac{2}{3}$ if and only if there exists a directed graph of indegrees and outdegrees 2 containing the cycle $1 \cdot 2 \cdot 3 \cdots n-1 \cdot n \cdot 1$ as well as a length-two path from i to $i+1$ for each $i<n$ and also from n to 1. If no such digraph exists, the value of the problem is at least $\frac{3}{4}$. Moreover, it can be shown that such a digraph exists only when n is odd. Further, if n is odd the digraph is unique and there is a unique way of routing the commodities to achieve value $\frac{2}{3}$.

In summary,

- If n is odd, the value of the problem is $\frac{2}{3}$ and there is a unique optimal solution. All feasible solutions which are different from the optimal in the integer variables have value at least $\frac{3}{4}$.
- If n is even, the value of the problem is at least $\frac{3}{4}$.

This type of problem would be difficult for any algorithm because the instances for n and for $n+1$ are very similar. Moreover, for n odd, until we find the optimal solution we will be more than 10% away from the optimum. While examples of this sort are admittedly contrived, we can expect subtle combinatorial difficulties to arise from the pattern of demand values.

With regards to previous work on valid inequalities for uncapacitated fixed-charge network flow problems, besides the dicut collection inequalities of Rardin and Wolsey [15] mentioned above, which subsume the 'basic network inequalities' of Van Roy and Wolsey [17], very little appears to be known concerning polyhedral structure, especially facets, except in special cases, such as the economic lot-sizing problem. Balakrishnan, Magnanti and Wong [1] developed a computationally efficient procedure for solving the LP relaxation of the disaggregated formulation for uncapacitated fixed-charge network flow problems without any side constraints. The disaggregated formulation for this problem turned out to be very tight.

2. Valid inequalities

Next we describe some of the inequalities that have proved useful towards obtaining good bounds. We are dealing with the aggregated formulation, so that a commodity will be identified with its source node. In what follows we will say that a digraph is *of degree* 2 if

the indegree and outdegree of every vertex is 2. For completeness, we state the following result, which is implied by some of the results presented below. A complete proof of it appears in [7].

Lemma 2.1. *The dimension of* ICONG(2) *is* $n^2(n-1)-(2n-1)+1$. □

This lemma simply states that the dimension of ICONG(2) is precisely equal to the number of variables minus the rank of the formulation, that is, there are no additional implied equations. To see this, note that the expression above can be rewritten as $n(n(n-1)-(n-1))+n(n-1)-(2n-1)+1$. The first term here is n times the dimension of a one-commodity network flow polyhedron in a digraph with $n(n-1)$ edges. The next two terms correspond to the x variables. Here note that the degree constraints in ICONG(2) describe a transportation problem in a graph with $2n$ nodes and $n(n-1)$ edges. Lastly, the variable z (which need not satisfy any inequality tightly) contributes one additional unit of dimension.

2.1. A basic facet

For any commodity k and subset S of nodes write $t^k(S)=\sum_{i\in S} t_{ki}$. The main result in this section is:

Theorem 2.2. *Let k be a commodity and $S\subset\{1,\ldots,n\}\setminus k$. Write $T=\{1,\ldots,n\}\setminus(S\cup k)$. Then inequality*

$$\sum_{i\in T, j\in S} f_{kij} \geqslant \left(1-\sum_{i\in S} x_{ki}\right) t^k(S) \tag{7}$$

is a facet of ICONG(2) *provided* $2\leqslant |S|\leqslant n-3$, $t^k(S)>0$ *and* $t^k(T)>0$. □

For a full proof of this theorem refer to [7]. Here we will provide an outline of the proof. We denote B^k the set of k-vectors all of whose coordinates are 0 or 1. A digraph is called *strong* if it contains a directed path from each vertex to every other vertex. If $x\in B^{n(n-1)}$ we let $G[x]$ be the digraph whose edge set has incidence vector x. For any subset K of commodities, let P^K denote the convex hull of points satisfying the following constraints:

$$0\leqslant f_{kij}, \quad \text{all } k\in K \text{ and } i\neq j, \quad x_{ij}\in\{0,1\}, \quad \text{all } i\neq j,$$

$$\sum_{j\neq i} x_{ij}=2 \quad \text{for all } i, \tag{8}$$

$$\sum_{j\neq i} x_{ji}=2 \quad \text{for all } i, \tag{9}$$

$$x_{ij}=1 \text{ if } f_{kij}>0 \quad \text{for all } k\in K \text{ and } i\neq j, \tag{10}$$

$$\sum_{j\neq i} f_{kji}-\sum_{j\neq i} f_{kij}=t_{ki} \quad \text{for all } i \text{ and } k\in K, \tag{11}$$

$G[x]$ strong . (12)

In essence this is the formulation for ICONG(2) restricted to commodities in K, without the variable z, and with the added restriction that the digraph we use must be strong. We abbreviate $P^{\{k\}}$ as P^k. A proof of the following result appears in [7].

Lemma 2.3. *Inequality* (7) *defines a facet of* P^k *if the conditions in Theorem* 2.2 *hold. Moreover,* $\dim P^k = 2n(n-1) - (n-1) - (2n-1)$. □

Using Lemma 2.3 we can now complete the proof of Theorem 2.2. In what follows we assume that commodity k and set S satisfy the conditions of Theorem 2.2. Points in P^k will be given as pairs (x, f_k).

Lemma 2.4. *Let* $h \neq k$ *be a commodity. There exist affinely independent points* $(x^i, f^i_k) \in P^k$, $1 \leqslant i \leqslant d$ *(for some* d*) satisfying* (7) *with equality, such that:*

(1) *For* $1 \leqslant i \leqslant d$, $G[x^i]$ *is strong and of degree* 2.

(2) *For* $1 \leqslant i \leqslant d$, $G[x^i]$ *contains an arborescence* A^i *rooted at* h*, such that, for every edge* $e \notin A^1$ *there exists* $1 \leqslant i \leqslant d$ *with* $e \in G[x^i] \setminus A^i$ *but* $e \notin G[x^j]$ *for each* $j < i$.

(3) *For* $2 \leqslant i$, $G[x^i]$ *contains an edge not in* $\bigcup_{j<i} G[x^j]$.

Proof. To simplify notation, assume $k = n$, $h \neq n-1$, $S = \{1, \ldots, s\}$ and $T = \{s+1, \ldots, n-1\}$. Let H be the digraph consisting of the cycle $1 \cdot 2 \cdots s \cdot n \cdot n-2 \cdot n-3 \cdots s+1 \cdot 1$ and the edge $(n, n-1)$. We have

(a) If $G \supseteq H$ is a strong digraph of degree 2 then there is a point $(x, f) \in P^n$ satisfying (7) with equality, such that $G = G[x]$.

(b) For every edge $e = (u, v) \notin H$ with $u \neq n$, there is a strong digraph G of degree 2 including $H \cup e$.

To see that (a) holds, notice that in the arborescence $H \setminus (s, n)$ all vertices of T are reached from n before the vertices of S, and that if G is as in (a) and $G = G[x]$ (for some $\{0, 1\}$-vector x) then $\sum_{i \in S} x_{ni} = 0$. To see that (b) holds, notice that if we add to H the edge $(n-1, n)$ and the cycle $C = s \cdot s-1 \cdots 2 \cdot 1 \cdot n-1 \cdot s+1 \cdots n-3 \cdot n-2 \cdot s$ we obtain a strong digraph of degree 2. So if e is in this digraph we are done. If not, then it is easy to see how to break up the cycle C to conclude that (b) holds. Let A be the arborescence obtained by removing from H the edge entering h. We can construct points $(x^i, f^i_k) \in P^k$, such that conditions (1) and (3) hold, and condition (2) holds for each edge e whose tail is not n (using $A^i = A$). It is easy to see that the same can be done for each edge of the form (n, t). The fact that the points (x^i, f^i_k) are affinely independent follows from (2). This concludes the proof of the lemma. □

Lemma 2.5. *For any* $h \neq k$ *inequality* (7) *defines a facet of* $P^{\{k,h\}}$.

Proof. Let F be the face of $P^{\{k,h\}}$ induced by (7). To prove this lemma, we will construct $\dim P^k + n(n-1) - (n-1)$ affinely independent points in F (thereby also showing that

$\dim P^{\{k,h\}} = \dim P^k + n(n-1) - (n-1)$. The result will follow from this because the right-hand side in this expression is certainly an upper bound on $\dim P^{\{k,h\}}$). For convenience, denote $U = \dim P^{\{k,h\}} = \dim P^k + n(n-1) - (n-1)$.

For $1 \leqslant j \leqslant U$ the jth point we will construct will be denoted by v_j. Let d be the quantity produced by Lemma 2.4. The points we construct will be of two types:

Type 1. For $1 \leqslant i \leqslant d$, let $(x^i, f^i_k) \in P^k$, be the points produced by Lemma 2.4. For $1 \leqslant i \leqslant d$, let $C(i) = G[x^i] \setminus (A^i \cup_{j<i} G[x^j])$, and $c(i) = |C(i)|$, and write $G[x^i] = G^i$. For each $1 \leqslant i \leqslant d$, we will construct a set $F(i)$ of $1 + c(i)$ points in F, each of them having projection (x^i, f^i_k) in the (x, f_k) space. By construction, this will yield

$$\sum_{i=1}^{i=d} (1 + c(i)) = d + \sum_{i=1}^{i=d} c(i) = d + n(n-1) - (n-1) \tag{13}$$

$$= U - (\dim P^k - d) \tag{14}$$

points.

Type 2. If $d < \dim P^k$, for each j with $U - (\dim P^k - d) \leqslant j \leqslant U$ we construct an additional point v_j with the following property. If w_j is the projection of v_j in the (x, f_k) space, then we require that the family of points

$$\{(x^i, f^i_k): 1 \leqslant i \leqslant d\} \cup \{w_j: U - (\dim P^k - d) \leqslant j \leqslant U\} \tag{15}$$

be affinely independent.

First we handle the Type 1 points. Choose a fixed i, $1 \leqslant i \leqslant d$. For simplicity, when describing the points in $F(i)$ we will only give their f_h coordinates. Let g^i be a circulation flow (in the space of commodity h), such that for any edge $e \in G^i$, $g^k_e > \sum_j t_{hj}$. Then:

- We obtain one point of the form $g^i + a^i$, where a^i is the flow vector obtained by routing commodity h on A^i (this will be called the *first* point in $F(i)$).
- For each $e \in C(i)$ we obtain an additional point of the form $g^i + a^i + \theta^i_e$, by pushing a small amount of flow along the cycle obtained by adding e to A^i.

Proceeding in this manner we produce $1 + c(i)$ points as desired.

Next, we construct the Type 2 points. By Lemma 2.3 it is clear that we can find points $w_j \in P^k$ satisfying (7) with equality so that the affine independence condition in the definition of Type 2 points is satisfied. By definition of P^k, the digraph corresponding to each of these points is strong, and so we can route commodity h arbitrarily, yielding points $v_j \in F$ as desired.

We claim that the points v_j, $1 \leqslant j \leqslant U$ are affinely independent. For suppose that for some coefficients α, $\sum_j \alpha_j v_j = 0$. By construction of the Type 2 points, we conclude $\alpha_j = 0$ for $j > d$. So we just have to show that the vectors in $\bigcup_{i \leqslant d} F(i)$ are affinely independent. For $1 \leqslant i \leqslant d$, subtract the first vector in $F(i)$ from the other vectors in $F(i)$, obtaining a family $F'(i)$. By construction, the last $c(i)$ vectors in $F'(i)$ will have projection $(0, 0)$ in the (x, f_k) space, but for each $e \in C(i)$ precisely one of these vectors will have a positive coordinate on edge e. Recall that for any such e we have $e \notin G^j$ for every $j < i$ (property (2) of Lemma 2.4). So if there is a nontrivial linear combination of the vectors in $\bigcup_{i \leqslant d} F'(i)$ that adds to zero, then only the first vector from each $F(i)$ can have a nonzero coefficient. But these are

affinely independent because their projections in the (x, f_k) space are, again by Lemma 2.4. This concludes the proof. □

By applying the same technique as in the previous lemma, one commodity at a time, we will conclude that inequality (7) defines a facet of $P^{\{1,\dots,n\}}$. That is, we will construct a family $\mathfrak{T}$ of

$$\dim P^k + (n-1)(n(n-1)-n+1) = n(n+1)(n-1) - n(n-1) - (2n-1)$$

affinely independent points in $P^{\{1,\dots,n\}}$, each satisfying (7) with equality. Each of the points of $\mathfrak{T}$ yields a point in ICONG(2) by setting $z = \max_{ij} \Sigma_k f_{kij}$. Consequently, (7) defines a facet of ICONG(2) if we can find one additional (affinely independent) point. This is easy: we simply take one of the points in $\mathfrak{T}$ and set the z-coordinate larger than $\max_{ij} \Sigma_k f_{kij}$. This concludes the proof of Theorem 2.2, modulo the proof of Lemma 2.3. □

Notes

(1) In the proof of Lemma 2.5 as given above, we need to allow positive flows of commodity h on edges entering h. On the other hand, when solving problem ICONG(2) we can set all variables of the form f_{hih} to 0. The proof of Theorem 2.2 can be adapted to take this into account.

(2) Given a facet of P^k, under what conditions is it also a facet of ICONG(2)? Of course one expects that this is always the case. It can be shown that if in the definitions of all the polyhedra above, the degree equations $\Sigma_j x_{ij} = \Sigma_j x_{ji} = 2$ are replaced with inequalities ($\leqslant 2$) the corresponding result holds.

One possible variant of inequality (7) is the following. Let k, S and T be as above, and let $U \subset S$. Then

$$\sum_{i \in T, j \in S} f_{kij} + t^k(S) \sum_{i \in S \setminus U} x_{ki} + \sum_{i \in U} f_{kki} \geqslant t^k(S) \tag{16}$$

is valid. However, computationally it has usually been the case that the variable upper-bound inequalities are tight for a given commodity k and edges (k, i). As a consequence, f_{kki} tends to be close to $(t^k(S) + t^k(T))x_{ki}$ with the result that (7) dominates (16).

It is difficult to separate over inequalities (7) (or in general, over dicut collection inequalities, of which (7) is a special case). Moreover, we remind the reader of the result in [15], that using all dicut collection inequalities yields the projection of the disaggregated formulation. Further, as we saw in the previous section, this formulation cannot be expected to be substantially stronger. On the positive side, it can be shown that (under fairly general conditions) (7) remains facet-defining even after fixing some of the x_{ij} variables.

To close this section, we point out the following (perhaps curious) result.

Proposition 2.6. *Let* $P = \text{conv}\{x \in B^{n(n-1)}: G[x]$ *of degree* $2\}$ *and* $Q = \text{conv}\{x \in B^{n(n-1)}: G[x]$ *is strong and of degree* $2\}$. *For* $n \geqslant 4$, $\dim P = \dim Q$. □

2.2. Source inequalities

In this section we consider polyhedra related to variable upper-bound flow models; see [14] and [13], Sections II.2.4 and II.6.4.

For any fixed $F>0$ and integer $n>2$, let $P_n(2F)$ be the convex hull of points $(x, f, z) \in \mathbb{R}^{2n+1}$ (where $x \in \mathbb{R}^n, f \in \mathbb{R}^n$ and $z \in \mathbb{R}$) satisfying:

$$\sum_{i=1}^{n} f_i = 2F, \tag{17}$$

$$\sum_{i=1}^{n} x_i = 2, \tag{18}$$

$$x_i = 1 \quad \text{if } f_i > 0, \quad i \in \{1, \ldots, n\}, \tag{19}$$

$$f_i \leqslant z, \quad i \in \{1, \ldots, n\}, \tag{20}$$

$$0 \leqslant f_i, \quad i \in \{1, \ldots, n\}, \tag{21}$$

$$x_i \in \{0, 1\}, \quad i \in \{1, \ldots, n\}, \tag{22}$$

In other words: we have a supply of $2F$ units of flow, that must be pushed through two edges chosen from among n candidates. The largest of all flows is (a lower bound for) z.

In what follows $e^i \in \mathbb{R}^n$ will be the ith unit vector, and $e^{ij} = e^i + e^j$. We have:

Lemma 2.7. *The dimension of $P_n(2F)$ is $2n-1$ for $n>2$.*

Proof. Consider the following $2n$ points in $P_n(2F)$: (e^{1i}, Fe^{1i}, F) for $2 \leqslant i \leqslant n$, $(e^{1i}, 2Fe^1, 2F)$ for $2 \leqslant i \leqslant n$, $(e^{12}, 2Fe^2, 2F)$, and (e^{23}, Fe^{23}, F). The $2n \times (2n+1)$ matrix formed by these vectors is

$$\left(\begin{array}{c|ccc|c|ccc|c}
1 & & & & F & & & & F \\
\vdots & & I_{n-1} & & \vdots & & FI_{n-1} & & \vdots \\
1 & & & & F & & & & F \\
\hline
1 & & & & 2F & & & & 2F \\
\vdots & & I_{n-1} & & \vdots & & 0 & & \vdots \\
1 & & & & 2F & & & & 2F \\
\hline
1 & 1\,0\,0 & \cdots & 0 & 0 & 2F\,0\,0 & \cdots & 0 & 2F \\
0 & 1\,1\,0 & \cdots & 0 & 0 & F\,F\,0 & \cdots & 0 & F
\end{array}\right) \tag{23}$$

and is seen to have full row rank. Because of equations (17) and (18) the result follows. □

In what follows, for any vector v and $S \subseteq \{1, \ldots, n\}$, $v(S) = \sum_{i \in S} v(i)$. Consider the following inequalities:

$$z - f(S) + Fx(S) \geqslant F, \tag{24}$$

for all $S \subset \{1, \ldots, n\}$ with $1 \leqslant |S| < n$,

$$z - 2f(S) + 2Fx(S) - f_i \geqslant 0\,, \tag{25}$$

for all $S \subset \{1, \ldots, n\}$ with $1 \leqslant |S| \leqslant n-2$ and $i \notin S$, and

$$f_i \leqslant 2Fx_i \tag{26}$$

for all $i \in \{1, \ldots, n\}$. Below we will show that these are facet defining. Inequalities (24) and (25) strengthen the inequality $z \geqslant F$, which is valid because $2F$ units of flow must be routed on 2 edges. Typically, the linear programming formulation (once we replace (20) by an appropriate variable upper bound inequality) will 'cheat' by spreading the $2F$ units of flow over more than 2 terms somewhat unevenly, while keeping z small. That is to say, for some indices i we may have $f_i > Fx_i$. Inequalities (24) and (25) cut off such fractional points. It would be interesting to obtain these facets using the MIR procedure ([13]).

Lemma 2.8. *Inequalities* (24)–(26) *define facets of* $P_n(2F)$.

Proof. Consider inequality (24) for a given subset S. If $x(S) = 0$ or if $x(S) = 2$ the inequality is valid, since in either case $f(S) - Fx(S) = 0$. If $x(S) = 1$, say $x_i = 1$ for some $i \in S$, then $z \geqslant f_i = f_i - F + F = f(S) - Fx(S) + F$. To see that (24) is facet inducing, assume w.l.o.g. that $S = \{1, \ldots, |S|\}$, and that $|S| \leqslant n-2$ (the case $|S| = n-1$ is similar). Write $s = |S|$. Consider the $2n-1$ points (e^{1i}, Fe^{1i}, F) for each $i \notin S$, $(e^{i,s+1}, Fe^{i,s+1}, F)$ for each $i \in S \setminus 1$, $(e^{1i}, 2Fe^1, 2F)$ for each $i \notin S$, $(e^{i,s+1}, 2Fe^i, 2F)$ for each $i \in S \setminus 1$ and $(e^{s+1,s+2}, Fe^{s+1,s+2}, F)$. Each of these points satisfies (24) with equality, and together they make up the matrix

$$\left(\begin{array}{c|ccc|ccc|c|ccc|ccc|c}
1 & 0 & \cdots & 0 & & & & F & 0 & \cdots & 0 & & & & F \\
\vdots & \vdots & \cdots & \vdots & & I_{n-s} & & \vdots & \vdots & \cdots & \vdots & & FI_{n-s} & & \vdots \\
1 & 0 & \cdots & 0 & & & & F & 0 & \cdots & 0 & & & & F \\
\hline
0 & & & & 1\,0 & \cdots & 0\,0 & 0 & & & & F\,0\,0 & \cdots & 0 & F \\
\vdots & & I_{s-1} & & \vdots & \cdots & \vdots & \vdots & & FI_{s-1} & & \vdots & \cdots & \vdots & \vdots \\
0 & & & & 1\,0 & \cdots & 0\,0 & 0 & & & & F\,0\,0 & \cdots & 0 & F \\
\hline
1 & 0 & \cdots & 0 & & & & 2F & 0 & \cdots & 0 & 0\,0\,0 & \cdots & 0 & 2F \\
\vdots & \vdots & \cdots & \vdots & & I_{n-s} & & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\
1 & 0 & \cdots & 0 & & & & 2F & 0 & \cdots & 0 & 0\,0\,0 & \cdots & 0 & 2F \\
\hline
0 & & & & 1\,0 & \cdots & 0\,0 & 0 & & & & 0\,0\,0 & \cdots & 0 & 2F \\
\vdots & & I_{s-1} & & \vdots & \cdots & \vdots & \vdots & & 2FI_{s-1} & & \vdots & \cdots & \vdots & \vdots \\
0 & & & & 1\,0 & \cdots & 0\,0 & 0 & & & & 0\,0\,0 & \cdots & 0 & 2F \\
\hline
0 & 0 & \cdots & 0 & 1\,1 & \cdots & 0\,0 & 0 & 0 & \cdots & 0 & F\,F\,0 & \cdots & 0 & F
\end{array}\right) \tag{27}$$

from which it is easy to argue that the points are affinely independent, as desired. The proofs that (25) and (26) are facet inducing are similar. □

Theorem 2.9. *The facet defining inequalities* (24), (25) *and* (26), *together with* (17), (18), (21) *as well as the bounds* $0 \leqslant x_i \leqslant 1$ *(for every* i*) yield the polyhedron* $P_n(2F)$.

Proof. Refer to [7]. □

In terms of problem ICONG(2), inequalities (24) and (25) and (26) cannot be used directly as cutting planes (for any given node k) because the value of F is not fixed. But we obtain valid inequalities by (1) replacing $x(S)$ with $2-x(V\backslash S)$, (2) replacing F in all linear terms with the total flow leaving k ($\sum_{i,h} f_{hki}$), and (3) in the terms of the form $-Fx(S)$ by replacing F with a lower bound. Such a lower bound is available: half of the sum of all demands with source node k. The resulting inequalities can be improved if an upper bound on F is available, or better, if an upper bound u on the individual flows f_i is available. Such an upper bound is frequently available if we have one the value of the instance of ICONG(2). A different perspective on this is the following. Notice that in the preceding proofs we constructed points with $f_i = 2F$ for certain indices i. That is, all flow is routed on one edge. In computation, we would expect that this might not happen, i.e., flow would be split at least among two edges so as to keep z small. This suggests that the above inequalities may not necessarily be active, especially after a few rounds of cutting planes. What might be needed, instead, are inequalities that handle cases where z is larger than F but much smaller than $2F$.

To that effect we first consider, for given $0 < F \leqslant u$ the polyhedron $Q_n(2F, u)$ obtained by replacing (in the formulation of $P_n(2F)$) (19) with $f_i \leqslant u x_i$. This is of interest when $F < u$. By adapting the proof of Lemma 2.8 one has:

Lemma 2.10. *For all* $0 < F < u$ (24) *and* (25) *induce facets of* $Q_n(2F, u)$. □

We omit this proof because it is very similar to the one for Lemma 2.8. All one has to do is replace the points with $z = 2F$ with points where $z = u$, which is possible because $F < u$.

There is another way of improving on the above polyhedra, which arises by stating that we have to push *at least* $2L$ units of flow, rather than exactly $2F$ units, for some L. More precisely, consider for fixed $0 < L < u$ the polyhedron $R_n(2L, u)$ obtained by replacing (in the formulation of $Q_n(2L, u)$) (17) with $\sum_i f_i \geqslant 2L$. Computationally, this has been most useful when $L < u < 2L$, in which case the variable upper-bound inequalities become active. One can obtain facets for $R_n(2L, u)$ by starting with facets for $Q_n(2L, u)$ and 'lifting' them so as to be tight for at least one point in $Q_n(2u, u)$. In this way one has (where $V = \{1, \ldots, n\}$):

Lemma 2.11. *For every* $0 < L < u$ *the following inequalities define facets of* $R_n(2L, u)$:

$$z \geqslant \tfrac{1}{2} f(S) - \tfrac{1}{2} f(V\backslash S) - Lx(S) + 2L\,, \tag{28}$$

for all $S \subset V$ *with* $2 \leqslant |S| \leqslant n-1$, *and*

$$z \geqslant 2(Lx(S) - f(S)) + f_j + \min\left\{1, \frac{u}{2(u-L)}\right\}(f(V) - 2L)\,, \tag{29}$$

for all $S \subset V$ *with* $2 \leqslant |S| \leqslant n-2$ *and* $j \in S$, *and*

$$z \geqslant f_j - Lx_j + L\,, \tag{30}$$

$$f_j \geqslant (2L-u)x_j\,, \tag{31}$$

$$z \geqslant f(V) - f_j - u(1-x_j)\,. \qquad \square \tag{32}$$

We obtained these inequalities using the lifting procedure described above, which guarantees that they are facets. Inequality (28) is especially active when we have a set S with $x(S)=1$ (or close to it) and $f(S) > f(V\backslash S)$. The inequality says that in an integral solution *the* edge used in S must carry flow at least $\frac{1}{2}f(S) - \frac{1}{2}f(V\backslash S) + L$ (which is equal to $2L - f(V\backslash S)$, thus showing validity), a value strictly larger than L. As a result, the inequality tends to cut off fractional points with slight imbalances among the variables f_i. The other inequalities above have similar interpretations.

2.3. Flux inequalities

In [9] the following procedure was used to obtain a lower bound for z in an instance of ICONG(2): if F^* is a lower bound on $\sum_k \sum_{i,j} f_{kij}$ then clearly $z \geqslant F^*/2n$. To compute such a lower bound F^*, we denote by F^1 the sum of the $2n$ largest demands, and for $d > 1$, F^d the sum of the $2^{d-1}n + 1$ through $2^d n$ largest traffic demands. Then

$$F^* = \sum_d dF^d \tag{33}$$

is a valid lower bound, since in a digraph with outdegrees at most 2, at most $2^d n$ pairs of vertices are at distance d, for any d. This lower bound tends to be weak because it averages over all commodities. In particular, for any given node i at most 2^d nodes can be at distance d from i, yet the quantity F^d may include more than 2^d traffic elements, all with source node i. A heuristic way of improving the lower bound is given in [9].

Equation (33) can be strengthened if we adapt it to work one commodity at a time. In what follows we consider a fixed commodity (i.e., source node) k. For a subset of edges S, write $f_k(S) = \sum_{i,j \in S} f_{kij}$. We call $f_k(S)$ the *flux* of commodity k on S. Denote by E the set of all edges, i.e., the set of all ordered pairs of nodes. Then

$$f_k(E) \geqslant \sum_d dT^d\,, \tag{34}$$

where T^1 is the sum of the two largest demands with source k, and for $d \geqslant 1$, T^d is the sum of the $2^{d-1}+1$ through 2^d largest such demands. This inequality can be strengthened by noting that for any node $i \neq k$, if i is at distance one from k then i contributes t_{ki} to $f_k(E)$ (and otherwise i contributes at least $2t_{ki}$). So we can write

$$f_k(E) \geqslant \sum_{i \neq k} (2-x_{ki})t_{ki} + \sum_{d>2} \left[(d-2) \sum \{t_{ki}\colon i \text{ at distance } d \text{ from } k\} \right] \tag{35}$$

and so

$$f_k(E) \geqslant \sum_{i \neq k} (2 - x_{ki}) t_{ki} + \sum_{d>2} (d-2) T^d . \tag{36}$$

This inequality can further be strengthened as follows. For any d let $V(d)$ be the set of nodes i such that t_{ki} is included in T^d. For any $d > 1$, let

$$s(d) = \sum_{2 \leqslant h < d} \min\{t_{ki} : i \in V(h)\} . \tag{37}$$

Then

$$f_k(E) \geqslant \sum_{i \neq k} (2 - x_{ki}) t_{ki} + \sum_{d>2} (d-2) T^d + \sum_{d>2} \sum_{i \in V(d)} (s(d) - (d-2) t_{ki}) x_{ki} \tag{38}$$

is a valid inequality. To see this, notice that if $x_{ki} = 1$ for some $i \in V(d)$ with $d > 2$, then the right-hand side of inequality (35) exceeds that of inequality (36) by at least $s(d) - (d-2) t_{ki}$.

Example 2.12. Suppose the nonzero demands t_{ki} are 100, 90, 75, 70, 68, 62 and 30, corresponding to nodes $i = 1, \ldots, 7$. Then inequalities (34), (36) and (38), respectively, are

$$f_k(E) \geqslant 830 , \tag{39}$$

$$f_k(E) \geqslant 1020 - 100x_{k1} - 90x_{k2} - 75x_{k3} - 70x_{k4} - 68x_{k5} - 62x_{k6} - 30x_{k7} , \tag{40}$$

$$f_k(E) \geqslant 1020 - 100x_{k1} - 90x_{k2} - 75x_{k3} - 70x_{k4} - 68x_{k5} - 62x_{k6} + 2x_{k7} . \tag{41}$$

In the rest of this section we discuss valid inequalities involving quantities $f_k(S)$ as well as further ways of strengthening (36) and (38). First we need an auxiliary result. The following inequality, while not globally valid, does not cut off at least one optimal integral solution; and at the same time it is fairly useful towards strengthening the formulation.

Proposition 2.13. *Let z^L be a known lower bound on the value of a problem. For any commodity k and node i the following inequality is valid, without loss of generality:*

$$f_{kki} \geqslant \min\{t_{ki}, z^L\} \, x_{ki} . \tag{42}$$

Proof. If $x_{ki} = 0$ validity is clear, and so assume $x_{ki} = 1$, and that $f_{kki} < t_{ki}$. In this case there is a path P from k to i carrying positive flow of commodity k. Since $f_{kki} < z$, we can reroute a small amount of flow from this path to the edge (k, i) (and if necessary reroute flow of other commodities from (k, i) to P) without increasing the value of z. □

There are many reasons why inequalities (34) or (38) may fail to be tight, and we discuss two of them next. Notice that for these inequalities to be tight or nearly so (at an integral solution) commodity k must be routed using a shortest path tree that is balanced. Such a tree will not exist if the graph contains certain subgraphs, which we might call

'obstructions'. For example, if the graph contains edges (k, i) and (i, k) for a given i, then there will be at most three vertices at distance 2 from k (rather than four) and this will affect the number of vertices at all distances $d \geqslant 2$ from k. Thus in principle inequality (34) can be strengthened as follows:

$$f_k(E) \geqslant \sum_d dT^d + \sum_{i,j} \alpha_{ij}(x_{ij} + x_{ji} - 1)^+ \tag{43}$$

for appropriate parameters α_{ij}. Similar inequalities can be used with other obstructions. By themselves, inequalities of this type are rather weak. However, it can be the case that one can argue that at least some number of obstructions *must* occur in any digraph of degree two. For example, it can be shown that for $n = 8$ at least one of three obstructions must occur: at least one pair of parallel edges (i, j), (j, i), or a triple of edges (i, j), (i, h), (h, j), or a quadruple (i, j), (i, h), (j, g), (h, g). This fact can be stated with an appropriate valid inequality. In conjunction with inequalities (43), inequalities of this type can have a powerful effect on the quality of the lower bound, especially near optimality. On the other hand, determining the numbers of obstructions that must occur is a very difficult combinatorial problem and so this technique can be used strictly in an ad hoc manner.

The other reason why inequalities (34), (38) can be weak can be handled computationally, and it occurs when some demand is large compared with some known upper bound on the value of the problem and with the other demands with same source. As a simple example, suppose that the (positive) demands with source k are: 200, 50, 30, 20, 14, 10, 8 corresponding to nodes 1, ..., 7 and that we already know that 188 is an upper bound on the value of the problem. Since t_{k1} is so large the linear program will usually set $x_{k1} = 1$, and so it is important to strengthen our inequalities in this case. But for (34) or (38) to be tight we would again need a balanced tree, i.e., one where the demands of at least two nodes are routed on edge $(k, 1)$. But $t_{k1} > 188$ and so less than t_{k1} units of flow are routed on this edge and the tree becomes unbalanced. If $x_{k1} = 1$ it is easily seen that

$$f_k(E) \geqslant 188 + 50 + 2(30 + 20) + 3(14 + 10 + 8 + 12) = 470 \tag{44}$$

(whereas the right-hand side of (34) is 422). If $x_{k1} = 0$ then $f_k(E) \geqslant 602$. This can be argued as follows. The inequality is clearly valid if node 1 is at distance three or greater from k. If it is at distance two, the fact that $z \leqslant 188$ implies that at most three nodes can be at distance two from k, and consequently

$$f_k(E) \geqslant 50 + 30 + 2(200 + 20 + 14) + 3(10 + 8) = 602 \tag{45}$$

as desired. As a result we have that $f_k(E) \geqslant 602 - 132x_{k1}$ is a valid inequality which will dominate (38), and which can itself be improved in the same way that (38) improves over (34).

The general procedure for handling this kind of situation is given next. We use the following notation. Let $D = \{d_1, d_2, \ldots\}$ be a list of numbers. If $\mathcal{Q} = \{Q_1, Q_2, \ldots\}$ is an ordered partition of D,

$$w^j(\mathcal{Q}) = \sum_{d_h \in Q_j} d_h\,, \tag{46}$$

and

$$W(\mathscr{C}) = \sum_j jw^j(\mathscr{C}) \ . \tag{47}$$

If $S \subseteq D$ we let $\mathscr{R}(D, S)$ be an ordered partition $\{D_1, D_2, \ldots\}$ of D such that $D_1 = S$, and for $j \geqslant 2$, D_j contains the $2(2^{j-2}-1)|S|+1$ through $2(2^{j-1}-1)|S|$ largest elements of $D \backslash S$, and we write

$$L(D, S) = W(\mathscr{R}(D, S)) \ , \tag{48}$$

$$L^i(D) = \min_{|S|=i}\{L(D, S)\} \ , \tag{49}$$

(note: the right-hand side of (34) equals $L^2(D)$, where D is the list of all demands with source k). In what follows, z^U and z^L denote known upper and lower bounds on the value of a given instance of ICONG(2). Now we have:

Lemma 2.14. *Suppose that for some commodity k and node $i \neq k$ we have $z^U < t_{ki}$. Let D be the list of demands with source node k, and let $\mathscr{C} = \{D_1, D_2, \ldots, D_m\}$ be the ordered partition that attains $L^1(D \backslash t_{ki})$. Define:*

$$l_d = \min\{t_{kj}: t_{kj} \in D_d\} \ , \tag{50}$$

for $d < m$ and also for $d = m$ if D_m has 2^{m-1} members; $l_m = 0$ otherwise. Also, write $l_m + 1 = 0$. Set

$$i^1 = \min\{s: 2 \leqslant s \leqslant m+1, \ l_s < t_{ki} - z^L\} \ , \tag{51}$$

$$i^2 = \min\{s: s \geqslant i^1, \ l_s \leqslant t_{ki} - z^U\} \ . \tag{52}$$

Let $\mathscr{P}$ be the ordered partition that attains

$$\min\{L(D, S): |S| = 2, \ t_{ki} \notin S\} \ . \tag{53}$$

Then:

$$\begin{aligned} f_k(E) \geqslant & \sum_j (2 - x_{kj}) t_{kj} + \left(\sum_{d>2} (d-2) w^d(\mathscr{C}) + \sum_{d \geqslant i^2} l(d) \right) x_{ki} \\ & + (i^2 - i^1 - 1)(t_{ki} - z^U) x_{ki} + (i^1 - 1)(t_{ki} x_{ki} - f_{kki}) \\ & + \left(\sum_{d>2} (d-2) w^d(\mathscr{P}) \right)(1 - x_{ki}) \end{aligned} \tag{54}$$

is valid.

Proof. Suppose first that $x_{ki} = 0$. Then the right-hand side of the inequality is $\sum_j (2 - x_{kj}) t_{kj} + \sum_{d>2} (d-2) w^d(\mathscr{P})$ and as in (38) it is easy to see that this is valid. Next, suppose $x_{ki} = 1$. By Proposition 2.13, $f_{kki} \geqslant z^L$. So, writing $\epsilon = t_{ki} - f_{kki}$,

$$\epsilon \leqslant t_{ki} - z^L \ . \tag{55}$$

Let h be such that $l_h \leqslant \epsilon$, and either $\epsilon < l_{h-1}$ or $h=2$. Since $\epsilon > 0$, $f_k(E) - f_{kki}$ is lower bounded by a quantity of the form (c.f. (48)) $L(D', \{t_{ks}\})$, where D' is obtained from D by replacing t_{ki} with ϵ and $s \neq i$. The assumption on ϵ implies that the partition that attains $L(D', \{t_{ks}\})$ can be obtained from $\mathscr{C}$ by putting ϵ in set h, and moving l_h to set $h+1$, l_{h+1} to set $h+2$, and so on. So we can write:

$$f_k(E) - f_{kki} \geqslant \sum_d dw^d(\mathscr{C}) + h\epsilon + \sum_{d \geqslant h} l_d \tag{56}$$

and arguing as when obtaining (36) from (34), this inequality can be strengthened as follows:

$$f_k(E) - f_{kki} \geqslant \sum_{j \neq i} (2 - x_{kj}) t_{kj} + \sum_{d>2} (d-2) w^d(\mathscr{C}) + h\epsilon + \sum_{d \geqslant h} l_d . \tag{57}$$

Further, $f_{kki} \leqslant z^U$ implies that $\epsilon \geqslant t_{ki} - z^U$, and since by definition of i^2 we have that $h \leqslant i^2$, we conclude

$$h\epsilon + \sum_{d \geqslant h} l_d \geqslant i^1 \epsilon + \sum_{d \geqslant i^2} l_d + (i^2 - i^1 - 1)(t_{ki} - z^U) x_{ki} . \tag{58}$$

Moreover, $i^1 \epsilon = (i^1 - 2)(t_{ki} x_{ki} - f_{kki}) + 2(t_{ki} - f_{kki})$ and so

$$\sum_{j \neq i} (2 - x_{kj}) t_{kj} + i^1 \epsilon = \sum_j (2 - x_{kj}) t_{kj} + (i^1 - 2)(t_{ki} x_{ki} - f_{kki}) + t_{ki} x_{ki} - 2 f_{kki} , \tag{59}$$

and combining (57), (58) and (59) one obtains the right-hand side of (54). □

Example 2.15. Let $D = \{300, 80, 70, 10, 10, 9, 7, 6, 5, 5, 5, 4, 3, 3, 2, 1\}$ (where $t_{k1} = 300$). Suppose $z^U = 291$ and $z^L = 262$. Then

$$\mathscr{C} = ((80), (70, 10), (10, 9, 7, 6), (5, 5, 5, 4, 3, 3, 2, 2), (1)) , \tag{60}$$

$l^2 = 10$, $l^3 = 6$, $l^4 = 2$, $l^5 = 0$, $i^1 = 2$, $i^2 = 3$, and

$$\mathscr{P} = ((80, 70), (300, 10, 10, 9), (7, 6, 5, 5, 5, 4, 3, 3), (2, 2, 1)) ,$$

$$\sum_{d>2} (d-2) w^d(\mathscr{C}) = 92 ,$$

$$\sum_{d \geqslant i^2} l_d = 8 . \tag{61}$$

The valid inequality is:

$$\begin{aligned} f_k(E) &\geqslant \sum_j (2 - x_{kj}) t_{kj} + 100 x_{k1} + 300 x_{k1} - f_{kk1} + (38 + 10)(1 - x_{k1}) \\ &= \sum_j (2 - x_{kj}) t_{kj} + 352 x_{k1} - f_{kk1} + 48 . \end{aligned} \tag{62}$$

A similar type of valid inequality can be used when $t_{ki} < z^U$, although the situation is more complicated in this case. We will need some further notation. Let D be a list of numbers, and consider an element $d_i \in D$ and a number $u \geqslant d_i$. Then we write

$$W(D, d_i, u) = \min_K \{\min_{A,B} \{L(A, d_i) + L^1(B)\}\}, \tag{63}$$

where

(1) K is any list obtained from D by replacing, for some $j \neq i$, d_j with two numbers of the form $d_j - \epsilon$ and ϵ, where $0 \leqslant \epsilon \leqslant d_j$, and

(2) A and B are lists whose union is K, such that $d_i \in A$ and the sum of the elements of A is at most u.

Example 2.16. Let $D = \{20, 20, 9, 8, 5\}$, $d_i = 20$ and $u = 23$. Then the minimum is attained by setting $K = \{20, 20, 9, 8, 3, 2\}$, $A = \{20, 3\}$ and $B = \{20, 9, 8, 2\}$.

Remark. The knapsack problem is a special case of that of computing quantities $W(D, d_i, u)$.

Inequality (34) can be strengthened as follows:

Proposition 2.17. *Consider an instance of* ICONG(2) *where z^U is a known upper bound on the value of the problem such that for some commodity k and node i, $t_{ki} \leqslant z^U$. Let D be the list of all demands with source node k. Then*

$$f_k(E) \geqslant W(D, t_{ki}, z^U) x_{ki} + (\min_S L(D, S))(1 - x_{ki}), \tag{64}$$

where the minimum is taken over all pairs S with $t_{ki} \notin S$. □

The proof of this proposition follows easily from the definitions given above. We also note that (64) can be strengthened by adding to the right-hand side terms involving variables x_{kj}, $j \neq i$.

Example 2.18. Let $D = \{300, 80, 50, 30, 25, 20, 16, 12\}$, $t_{ki} = 300$ and $z^U = 305$ (note that the sum of entries in D is 533). Then we have:

$$f_k(E) \geqslant 561 + \sum_j (1 - x_{kj}) t_{kj} - f_{kki} + 345 x_{ki}. \tag{65}$$

To see why this holds, notice that if $x_{ki} = 0$ the last two terms in the above inequality disappear and we obtain inequality (36). If $x_{ki} = 1$, then we can write $f_{kki} = 300 + \epsilon$, where $\epsilon \leqslant 5$. It is easy to verify that in this case $f_k(E) \geqslant 561 + \sum_j (1 - x_{kj}) t_{kj} - \epsilon + 25 + 20 = 561 + \sum_j (1 - x_{kj}) t_{kj} - f_{kki} + 345$.

The final type of flux inequalities that we use involve arbitrary node subsets S. Then it is possible to write an inequality of the form

$$\sum_{j \in S} f_{kij} \geqslant L + \sum_{j \in S} \alpha_{ij} x_{ij} + \sum_{i \notin S, j \in S} \beta_{ij} f_{kij}, \tag{66}$$

much in the same way that inequalities (34) and (36) were generated. It is computationally difficult to separate over inequalities of the form (66) and we have used them strictly in an ad hoc manner (see Section 3). Nevertheless, these inequalities are experimentally very useful when there are *clusters*: a cluster is a subset of nodes C such that the traffic demands are large amongst members of C, but low between C and its complement. In such a case we would use inequality (66) with $S = C$, or $S = C \setminus i$ for some $i \in C$.

2.4. Other inequalities

Here we describe various simple inequalities that appear useful in computation.

The first such inequality is (42) described in the previous section: for any commodity k and node i, $f_{kki} \geqslant \min\{t_{ki}, z^L\} x_{ki}$, for any lower bound z^L on the value of the problem.

The next class of inequalities are intended to strengthen the variable upper-bound inequalities (3). Consider any commodity k. In any feasible integer solution there will be two edges of the form (k, h), say (k, a) and (k, b) where $t_{ka} \leqslant t_{kb}$. Since we must route commodity k to satisfy the demand of each of a and b, as before one to argue that without loss of generality *both* (k, a) and (k, b) carry an amount of commodity k of value at least t_{ka}. Furthermore, for any edge (i, j), the flow of commodity k on this edge does not include any flow destined to satisfy the demand at i.

As a consequence, in inequality (3) we can set

$$M_{ij}^k = \sum_h t_{kh} - t_{ki} - \min_{h \neq i} \{t_{kh}\} . \tag{67}$$

A different (and usually more effective) way of tightening the variable upper bounds is given by the following inequality, which is one of the 'basic network inequalities' of [17]:

$$f_{kij} \leqslant t_{kj} x_{ij} + \sum_{h \neq i} f_{kjh} , \tag{68}$$

and whose validity is clear. In general, if S is a subset of nodes not containing node j, we can write

$$\sum_{i \in S} f_{kij} \leqslant \min\left\{1, \sum_{i \in S} x_{ij}\right\} t_{kj} + \sum_h f_{kjh} . \tag{69}$$

The following inequality can be used to tighten inequality (5) of the original formulation. For any pair, i, j we have:

$$z \geqslant \sum_k f_{kij} + z^L (1 - x_{ij}) . \tag{70}$$

Experimentally, this inequality has been very effective in terms of improving the lower bound z^L.

3. Computational results

In our computational experiments, the initial formulation consisted of the degree equations, the flow conservation equations, and inequalities (70) to measure the maximum load

(a lower bound on the value of the problem is always available as discussed before). The variable upper bounds strengthened as in (67) were used as cutting planes. Further, in many cases it proved advantageous to introduce new variables that aggregate others, in this way obtaining a sparser formulation. For example, for any edge (i, j) we can add the equation $F_{ij} = \sum_k f_{kij}$ and use F_{ij} instead of the right-hand side elsewhere. This strategy usually resulted in faster solving linear programs. The inequalities that we used were (7), those described in Lemma 2.11, (38), Proposition 2.13, (54), (64), (68), (69), as well as some of the inequalities strengthening (34), such as (66). These, although very strong, are difficult to separate and we typically added some of them to the formulation before running the automatic part of the algorithm.

We first applied the algorithm to the eight-node problems from [8] (also see [9]). As stated before, the resulting mixed-integer programs have 56 $\{0, 1\}$ variables, approximately 400 columns and start with approximately the same number of rows; typically we would end up with roughly 2000 rows. Each of the linear programs took (on the average) 7 to 10 seconds on a SPARC2 machine, using Cplex 2.0. We observed that the improvement in the lower bounds would taper off rather quickly after three or four iterations, and thus the overall algorithm would run in less than one minute. Some of our inequalities benefit from having good lower and upper bounds on the value of the problem. As a result, in all but one of the cases we ran the cutting-plane algorithm twice: once to get a lower bound (and an upper bound, see below) and then again with the bounds in place (in one of the cases the algorithm was run three times).

Table 2 summarizes our experience with the eight-node problems. For each problem, the column labeled 'LP-relax.' gives the LP-relaxation value of ICONG(2) including all variable upper-bound inequalities, the column labeled 'Strong ineqs.' gives the best lower bound obtained by the cutting-plane algorithm, 'Upper bound' is our upper bound and 'GAP' is the percentage gap between our bounds.

We were able to solve problems **ring** and **disconn** by running branch-and-bound on our extended formulation (with all variable upper-bound inequalities added). This required several tricks. For example, problem **disconn** has the following structure: the nodes are partitioned into two classes, such that traffic demands are large between nodes of the same class and small otherwise. The linear program will set the sum of the x variables going from one class to the other to a value between 1 and 2. So we can branch, by setting this sum of variables to 1, or 2, or at least 3. In each of these cases we can significantly strengthen the

Table 2
Performance of cutting-plane algorithm and heuristics for design-routing problem (* = optimally solved)

Problem	LP-relax.	Strong ineqs.	Upper bound	GAP (%)
uniform2	10.00	66.25	66.67	0.63
quasiunif1	9.92	58.63	61.83	5.46
quasiunif2	11.72	62.70	65.67	4.73
ring	33.81	113.74	124.00*	9.02
central	95.71	335.00	335.00*	0.00
disconn	62.50	255.80	275.40*	7.66

formulation by using inequalities of type (66) as well as others. Each of the resulting three problems had an LP-relaxation value (after running the cutting-plane algorithm) within 3% of the optimum, and each was separately solved to optimality using branch-and-bound (requiring a few hours in each case). A similar trick solved problem **ring**. Problem **central** is quite easy and the cutting-plane algorithm quickly found the integer optimum.

The upper bounds in Table 2 were obtained by us by branch-and-bound, except for the bound for problem **quasiunif2** which was obtained by W. Cook, by running his branch-and-bound algorithm on the extended formulation (which required several days of computing)[1]. To obtain upper bounds, the strategy suggested by L. Wolsey worked rather well for all problems: fix at 1 all $\{0, 1\}$-variables that are set that way by the cutting-plane algorithm, and run branch-and-bound on the remaining variables. This approach always found good solutions very quickly, sometimes in a few seconds. Typically these quick solutions were no more than 5% away from the optimum (and sometimes closer than that). We also used a (slower) heuristic to obtain better upper bounds that would run the cutting-plane algorithm, round to 1 some of the x_{ij} variables (based on their fractional value and using a randomized rule), and repeat until an integral solution was found. The overall process ran relatively quickly and we ran it several times for each problem.

In order to further test the strength of our lower bounds, we randomly generated sparse 20-node problems. In these problems the nodes are partitioned into clusters of four or five nodes each, such that traffic demands between two nodes are positive if and only if both nodes are in the same cluster. This structure makes it easy to find upper bounds – we solve the problem restricted to each cluster separately. For problems of this type, it is often the case that the optimal solution has edges between clusters and potentially these upper bounds could be crude. As discussed below, however, the linear programs that arose in these experiments were extremely difficult, and consequently we did not run branch-and-bound or our randomized heuristic on these problems, with the result that the simple upper bounds were all we had.

When running the cutting-plane algorithm, we employed a similar strategy as for the small problems, with one exception: the facet-defining inequalities (7) were used for (i.e., separated over) all small subsets, and heuristically for large subsets. We do not include a table of results for the randomly generated problems, but they can be summarized as follows. In all but one of these experiments the lower bounds we obtained were within 8 to 9% of our upper bounds. In one problem with clusters of size 5, the gap was approximately 11%.

As stated before, the linear programs arising here are quite difficult. Typically, the first linear program solved by the cutting-plane algorithm (i.e., one without the variable upper bounds) required on the order of 5 minutes (still using Cplex 2.0 on a SPARC2). After adding some cutting planes, the solution time would grow very quickly, sometimes to over one hour per LP. It is not clear precisely what makes these linear programs difficult. It is known that VUBs (variable upper bounds) make a formulation degenerate, and at first glance that could be a problem here. However, very few VUBs would be explicitly added to the formulation in the course of the algorithm. Moreover, the approach of randomly

[1] Optimally solved with value 65.67 by Bixby, Cook, Cox and Lee using eight parallel processors in three days.

Table 3
LP solution statistics

	Primal			Dual	
	Steepest edge	Devex	Red. cost	Steepest edge	Std. pricing
Iterations	3244(790)	8040(760)	25571(2573)	9288(2532)	18613(301)
Time	275.60	484.03	445.57	846.38	1684.88

perturbing the VUB coefficients did not seem to help at all. A similar approach would be to perturb the objective function and run a dual simplex algorithm: this helped if the perturbation was rather large, but undoing the perturbation proved just as hard as solving the problem itself.

Table 3 displays information concerning the solution of one of the initial linear programs using various pivoting strategies, now running Cplex 2.1 (still on a SPARC2).

This linear program has 8021 columns, 1320 rows and 32520 nonzeros. In the iterations row, the data in parentheses indicates Phase I pivots. Times are in seconds.

This data is typical (perhaps even a bit conservative) in that it shows that primal steepest edge pivoting is the best strategy (at least using our LP solver). The interior point code OB1 was also run on this problem, with negative results [12].

It is clear that in order to make branch-and-bound practicable (or even to run a traditional cutting-plane algorithm) a way must be found to speed-up the solution of these linear programs. (Presumably the LPs for problems on more than twenty nodes will be even more difficult.) One difficulty is that the 'good' cutting planes tend to be quite dense and of different types, and adding just a few of them can significantly change the linear program, so that starting from the previous optimal basis may not be helpful. This is an important area of work to be tackled.

4. Concluding remarks

There are several strategies for dealing with the problems discussed in the paper that appear promising, although whether they are computationally practicable is not clear.

One approach would be to selectively (and dynamically) disaggregate the problem (so that, for example, we may have several commodities corresponding to a given source node). Such a formulation could be strengthened in particular by using appropriate versions of our 'flux' inequalities (in fact, our inequalities (66) are an attempt to do something like that). Of course, this would have to be done with care so as to avoid blowing up the formulation, and we do not have an automatic criterion for doing this.

Another approach would be to develop a branch-and-cut algorithm with 'local' cuts at any node. In other words, it is usually the case for this problem that the formulation can be significantly strengthened at a branch. In particular, the flux inequalities and the variable upper-bound inequalities can be strengthened. However, we do not know how to do this

automatically and further, the cuts involved would not be globally valid and very different types of cuts would be used at different nodes.

One approach that worked, but appears very difficult to implement, is that of carefully adding new variables with a combinatorial interpretation. As mentioned above, we did this for variables of the form $(x_{ij}+x_{ji}-1)^+$ and in particular this worked well for problem **uniform2**: it narrowed the gap from 2.5% to less than 1%.

A final approach that does seem computationally efficient is the following. For a node i, denote $F(i)=\sum_{k,j} f_{kij}$. We know that $\sum_j t_{ij} \leqslant F(i)$ and if z^u is a known upper bound on the value of the problem some of our inequalities will force $F(i) \leqslant 2z^u$. One can proceed as follows: first partition the interval $[\sum_j t_{ij}, 2z^u]$ into a set of subintervals, each of a given size δ. Corresponding to any such subinterval we can branch, by forcing an inequality of the form $a \leqslant F(i) \leqslant a+\delta$. The rationale for this general approach is that the formulation can be tightened if we roughly know the value of $F(i)$. In particular, we may be able to reject a subinterval in one run of the cutting-plane algorithm. Further, in our experiments the interval $[\sum_j t_{ij}, 2z^u]$ was not very large, which would allow δ to be chosen rather small without generating many subintervals.

We have done some preliminary work in this area. However, we do not know how to automatically choose the quantities δ for an arbitrary problem (they should depend on the node i and should be dynamically adjusted) and also how to branch in an intelligent way.

Finally, it is clear that we must be able to solve the linear programs substantially faster.

Acknowledgements

The authors wish to thank W. Cook, L. Wolsey and R. Bixby for helpful advice, and an anonymous referee for a very thorough reading of the manuscript.

References

[1] A. Balakrishnan, T.L. Magnanti and R. Wong, "A dual-ascent procedure for large-scale uncapacitated network design," *Operations Research* 37 (1989) 716–740.

[2] S. Ceria, personal communication.

[3] S. Chopra, "Polyhedra of the equivalent subgraph and some edge connectivity problems," *SIAM Journal on Discrete Mathematics* 5 (1992) 321–337.

[4] W. Cook, personal communication.

[5] M.D. Grigoriadis and L.G. Khachiyan, "Fast approximation schemes for convex programs with many blocks and coupling constrains," Report DCS-TR-273, Department of Computer Science, Rutgers University (1991).

[6] M. Grötschel and C.L. Monma, "Integer polyhedra associated with certain network design problems with connectivity constraints," *SIAM Journal on Discrete Mathematics* 3 (1990) 502–523.

[7] O. Günlük, "Combinatorial optimization problems in lightwave networks," Ph.D. Thesis, Department of IEOR, Columbia University (1993).

[8] J.-F. Labourdette, "Rearrangeability techniques for multihop lightwave networks and application to distributed ATM switching systems," Tech. Report CU/CTR/TR 244-91-25, Center for Telecommunications Research, Columbia University (1991).

[9] J.-F. Labourdette and A. Acampora, "Partially reconfigurable multihop lightwave networks," *IEEE Transactions on Communication* 39 (1991) 1223–1230.
[10] F.T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos and S. Tragoudas, "Fast approximation algorithms for multicommodity flow problems," in: *Proceedings 23rd ACM Symposium on Theory of Computing* (May 1991) 101–111.
[11] T. Leong, P. Shor and C. Stein, "Implementation of a combinatorial multicommodity flow algorithms," in: *Proceedings DIMACS Implementation Challenge Workshop: Network Flows and Matching* (October 1991).
[12] I. Lustig, personal communication.
[13] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988).
[14] M.W. Padberg, T.J. Van Roy and L.A. Wolsey, "Valid linear inequalities for fixed charge problems," *Operations Research* 33 (1985) 842–861.
[15] R.L. Rardin and L.A. Wolsey, "Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network network flow problems," manuscript (1991).
[16] F. Shahrokhi and D.W. Matula, "The maximum concurrent flow problem," *Journal of the ACM* 37 (1990) 318–334.
[17] T.J. Van Roy and L.A. Wolsey, "Valid inequalities and separation for uncapacitated fixed charge networks," *Operations Research Letters* 4 (1985) 105–112.