Mathematical Programming 57 (1992) 375-414
North-Holland

Gainfree Leontief substitution flow problems

Robert G. Jeroslow†

College of Management, Georgia Institute of Technology, Atlanta, GA, USA

Kipp Martin

Graduate School of Business, University of Chicago, IL, USA, and Center for Operations Research and Econometrics, Universite Catholique de Louvain, Louvain-la-Neuve, Belgium

Ronald L. Rardin*

School of Industrial Engineering, Purdue University, West Lafayette, IN, USA

Jinchang Wang

College of Management, Georgia Institute of Technology, Atlanta, GA, USA

Received 31 May 1989
Revised manuscript received 8 January 1992

Dedicated to the memory of Robert G. Jeroslow

Leontief substitution systems have been studied by economists and operations researchers for many years. We show how such linear systems are naturally viewed as *Leontief substitution flow problems* on directed hypergraphs, and that important solution properties follow from structural characteristics of the hypergraphs. We give a strongly polynomial, non-simplex algorithm for Leontief substitution flow problems that satisfy a *gainfree* property leading to acyclic extreme solutions. Integrality conditions follow easily from this algorithm. Another structural property, *support disjoint reachability*, leads to necessary and sufficient conditions for extreme solutions to be binary. In a survey of applications, we show how the Leontief flow paradigm links polyhedral combinatorics, expert systems, mixed integer model formulation, and some problems in graph optimization.

Key words: Leontief matrices, linear programming, integer programming, network flows, polyhedral combinatorics, expert systems.

Introduction

Following Veinott (1968) we use the following definitions. The matrix A is *pre-Leontief* if each column contains at most one positive entry. The matrix A is *Leontief* if each column has exactly one positive element and there exists $\bar{x} \geqq 0$ such that $A\bar{x} > 0$.

Correspondence to: Prof. Ronald L. Rardin, School of Industrial Engineering, Purdue University, West Lafayette, IN 47907, USA.

† See Acknowledgement section.

* Research supported in part by the ONR (Office of Naval Research) under URI Grant number N00014-86-K-0689, and Center for Operations Research and Econometrics, Universite Catholique de Louvain.

This interesting class of matrices was first studied in the context of input-output analysis in economics. See Leontief (1951) and Dantzig (1955). Later these matrices were important within the context of dynamic programming and Markov processes (see, for example, Howard, 1960; Kemeny and Snell, 1960; Veinott, 1969b; and the more recent work of Erickson, 1978, 1988; and Rothblum and Whittle, 1982; on branching Markov decision chains). See also Veinott (1969a) for applications in operations management.

The algebraic properties of (*pre-*) *Leontief substitution systems* $Ax = b$, $x \geq 0$ with (pre-) Leontief matrices A and right-hand side $b \geq 0$ were thoroughly investigated in Veinott (1968) and Koehler, Whinston and Wright (1975). See the latter for an extensive set of references.

More recently, there has been a renewed interest in Leontief substitution systems because of the key role they play in polyhedral combinatorics, logic and expert systems. Erickson's (1978) dissertation took a Leontief approach to network flow problems with separable concave costs. Martin, Rardin and Campbell (1990) use properties of Leontief matrices in developing polynomial-size polyhedral descriptions of optimization algorithms on recursively defined graphs and facilities in series lot sizing problems. In the area of logic and expert systems Jeroslow and Wang (1989) characterize extreme solutions to a linear programming formulation of proofs over a Horn clause knowledge base. Ullman and Van Gelder (1988) study the problem of being able to guarantee when a Prolog-like evaluation of a set of logical rules terminates.

Our focus is the subclass of linear programs on Leontief and pre-Leontief constraint matrices satisfying a *gainfree* property. We begin by interpreting all linear programs over pre-Leontief systems as flow problems in a directed hypergraph. This interpretation is a natural extension of generalized networks to problems with more than one negative element in a column. In this context there is an equally natural notion of gain around a directed cycle. If no such gain cycle exists, Section 2 establishes that extreme point solutions contain no directed cycles. For extreme solutions that are *acyclic* in this sense, Section 3 provides simple algorithms to find optimal dual and primal solutions in time $O(mp)$, where m is the number of rows of A, and p is the number of nonzero entries in A. In Section 4 we show that this same acyclic structure of extreme solutions leads to total dual integrality and integer solutions under suitable hypotheses. Another structural property, *support disjoint reachability*, is seen to lead to a necessary and sufficient characterization of when the integer solutions must be binary. The remainder of the paper surveys results for different classes of applications. Some known properties are shown to follow from our theory of Sections 2–4, new results are proved, and some intriguing open questions are posed.

Because Leontief systems have been (re)discovered in so many settings over several decades, there is a wide, but not very unified literature. Some of our results, including most of those in Section 3, can be viewed as restatements in a new context of material found in earlier work of Veinott (1958), Erickson (1978, 1988) and others.

The major contribution of this paper is to introduce a directed hypergraph flow paradigm for studying Leontief systems that leads to both new structural results and an intuitive, unifying way of thinking about applications.

1. Directed hypergraph setting

Generalized networks are an extension of network flow problems by allowing the negative element in each column of the vertex-arc incidence matrix to be an arbitrary negative real number instead of -1. A pre-Leontief matrix further extends the vertex-arc incidence matrix of generalized networks by allowing more than one negative element in a column.

The corresponding graph structure is what we term a *Leontief directed hypergraph* $G=(V, H)$ on *vertices* in V and *hyperarcs* in H. Each hyperarc is an ordered pair (J, k), where $J \subseteq V$, $k \in V \backslash J$. Vertex k is the *head* of the hyperarc and J is the *tail set*. We allow *tailless or source* hyperarcs $(\emptyset, k)$ directed into k, and *headless or sink* hyperarcs $(J, \emptyset)$ directed out of tail set J. We assume throughout that the coefficient associated with the head of a hyperarc is $+1$. *Tail weights* $\{a_j[J, k] \mid j \in J\}$ give the magnitudes of corresponding negative coefficients at tails $j \in J$. Models can have several "parallel" hyperarcs with different tail weights connecting the same (J, k) pair. In the interest of notational simplicity, we assume each (J, k) pair indentifies a unique hyperarc. Parallel hyperarcs are easily made unique through the introduction of artificial vertices.

A *flow* on Leontief directed hypergraph $G=(V, H)$ is a set of hyperarc values $\{x[J, k] \mid (J, k) \in H\}$ conforming to conservation and nonnegativity constraints:

$$\sum_{(J,k)\in H} x[J, k] - \sum_{\substack{(J,l)\in H \\ k\in J}} a_k[J, l]x[J, l] = b_k \quad \text{for all } k \in V, \tag{1}$$

$$x[J, k] \geq 0 \quad \text{for all } (J, k) \in H. \tag{2}$$

Right hand sides b_k denote the net demands at vertices k.

Lemma 1.1. *For every $b \in \mathbb{R}^m$ and $m \times n$ pre-Leontief matrix A, there is a Leontief directed hypergraph G such that $Ax = b, x \geq 0$ corresponds to the hypergraph flow* (1)-(2).

Proof. Construct G by defining a vertex for every row of matrix A. For every column of A define the hyperarc (J, k) where the set J indexes the rows in which the column has a strictly negative element and k indexes the row in which the column has a positive element. If the column has no negative elements create the source hyperarc $(\emptyset, k)$; if the column has no positive elements create the sink hyperarc $(J, \emptyset)$. The b of the linear system defines the vector of net demands b_k. If column j of A

has a positive entry in row i, the tail weights are scaled absolute values $\{a_k[J, i] \stackrel{\text{def}}{=} -a_{k_j}/a_{i_j} \mid a_{k_j} < 0\}$. If there are no positive entries in column j, tail weights are simply $\{a_k[J, \emptyset] \stackrel{\text{def}}{=} -a_{k_j} \mid a_{k_j} < 0\}$. □

Associating unit flow costs $c[J, k]$ with hyperarcs $(J, k) \in H$, we define a *Leontief flow problem* as the linear program

$$\text{min} \quad \sum_{(J,l)\in H} c[J, l]x[J, l]$$

subject to (1)-(2).

That is, a Leontief flow problem is the generalization of the classic minimum cost network flow problem accommodating arcs with multiple tails and arbitrary tail weights. We use the term Leontief flow even when the constraint matrix of (1)-(2) is pre-Leontief.

All of our main results for Leontief flows address the Leontief substitution case where right hand side vector b is nonnegative. We term these *Leontief substitution flow problems* to indicate that $b \geqslant 0$ is assumed.

Veinott's (1968) classic work on Leontief substitution systems calls row i of a pre-Leontief matrix A *trivial* if for all $x \geqslant 0$ such that $Ax \geqslant 0$, the corresponding component of Ax is zero. Rows that are not trivial are *non-trivial*, and a matrix with exactly one positive entry per column is Leontief precisely when every row is nontrivial. Extending to the hypergraph setting, we term a vertex *trivial* if the corresponding row of the incidence matrix A is trivial, otherwise it is *nontrivial*. Thus a trivial vertex is one in which it is impossible to have a net positive demand.

We also employ a related notion of *degenerate* hyperarcs. A hyperarc (J, i) is degenerate, if and only if $x[J, i] = 0$ in every basic feasible solution to (1)-(2). After characterizing basic feasible solutions to Leontief substitution flow problems in Section 2, we show that if hyperarc (J, i) is nondegenerate in the $b \geqslant 0$ case then every vertex in J is nontrivial.

A *path* in Leontief directed hypergraph G from vertex v_1 to vertex v_{k+1} is defined by the non-null sequence $v_1 e_1 v_2 e_2 v_3, \ldots, e_k v_{k+1}$ whose terms are alternatively vertices and hyperarcs, with no vertex or hyperarc repeated, such that $e_i = (J_i, v_{i+1})$ and $v_i \in J_i$; or, $e_i = (J_i, v_i)$ and $v_{i+1} \in J_i$. A path from vertex v_1 to vertex v_{k+1} is a *directed path* if $e_i = (J_i, v_{i+1})$ and $v_i \in J_i$ for $i = 1, \ldots, k$.

A path from vertex v_1 to vertex v_{k+1} is a *cycle* if $v_1 = v_{k+1}$. A directed path from vertex v_1 to vertex v_{k+1} is a *directed cycle* if $v_1 = v_{k+1}$, and a Leontief directed hypergraph with no directed cycle is *acyclic*. A *hypertree* is a connected Leontief directed hypergraph which contains no cycles, and a *hyperarboresence* is a hypertree directed so that at most one hyperarc points into any vertex. A *hyperforest* is a Leontief directed hypergraph which contains no cycles.

The focus of this paper is on Leontief substitution flow problems over directed hypergraphs satisfying two newly isolated structural properties. One is the absence

of gain cycles. Let $v_1 e_1 v_2 e_2, \ldots, e_k v_{k+1}$ be a directed cycle where $v_1 = v_{k+1}$ and $e_i = (J_i, v_{i+1})$, $i = 1, \ldots, k$. The *gain* of this directed cycle is defined by

$$1 \Big/ \prod_{i=1}^{k} a_{v_i}[J_i, v_{i+1}]. \tag{3}$$

We term a Leontief flow problem defined on hypergraph G *gainfree* if the gain of every directed cycle in G is ≤ 1.

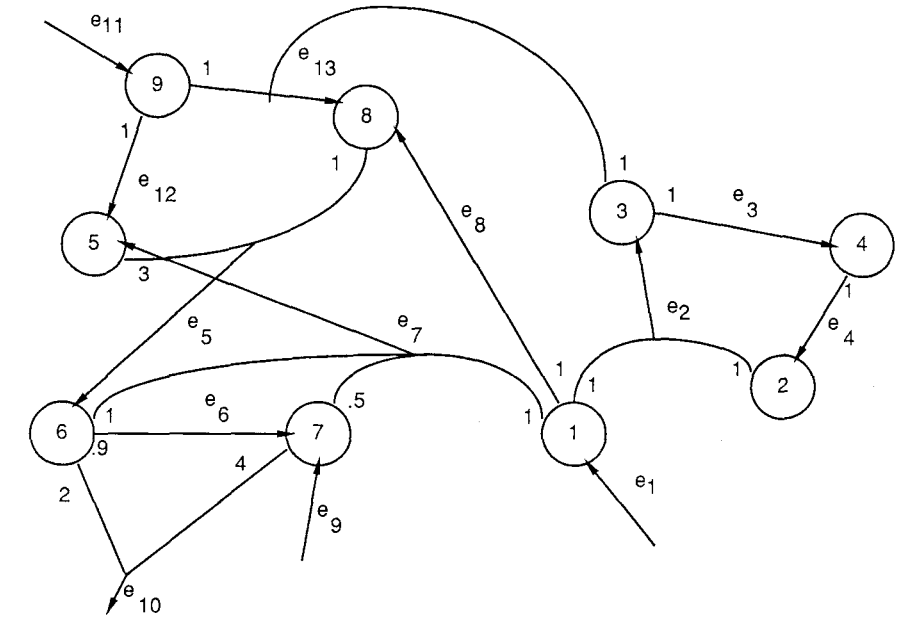
Our second property, *disjoint reachability*, relates to the absence of hypergraph objects we term *paracycles*. A paracycle is a subhypergraph consisting of directed paths $re_1 v_2 e_2, \ldots, e_k v_{k+1} gt$ and $rf_1 u_2 f_2, \ldots, f_l u_{l+1} gt$ disjoint except for the common root vertex r, final hyperarc $g = (J, t)$ and final vertex t. That is, a paracycle is a pair of disjoint paths from a common source to distinct tail vertices of the same hyperarc, plus the hyperarc itself. The structure is a "near" cycle in the sense that it would be a cycle if the last hyperarc g were replaced by arcs from v_{k+1} to t and u_{l+1} to t. Also, if vertex i has a directed path to vertex j within a hyperarborescence T, either the path is unique or T contains a paracycle.

Figure 1 illustrates these definitions. It shows a gainfree Leontief directed hypergraph on vertices $V = \{1, \ldots, 9\}$ and the corresponding pre-Leontief matrix A. In (J, k) notation hyperarc e_5 is $(\{5, 8\}, 6)$ with tail weights $a_5[\{5, 8\}, 6] = 3$, $a_8[\{5, 8\}, 6] = 1$. Source hyperarc e_9 is $(\emptyset, 7)$. The sequence $v_7 e_7 v_5 e_5 v_6 e_6 v_7$ is a directed cycle with gain $1/(0.5)(3)(0.9) < 1$. Hyperarcs e_5, e_{12} and e_{13} form a paracycle rooted at vertex 9, with disjoint paths to tail vertices 5 and 8 of hyperarc e_5. Even without the all negative column corresponding to arc e_{10}, the A matrix corresponding to this hypergraph is pre-Leontief and not Leontief because vertices v_2, v_3, v_4 are trivial.

2. Characterization of basic feasible solutions

Associated with every feasible solution to (1)-(2) is a *support hypergraph* which is the subhypergraph induced by all hyperarcs with a positive value in the solution. In this section we give a sufficient condition for when the support hypergraph of a basic feasible solution to a Leontief substitution flow problem is acyclic. This is obviously the case when hypergraph G defining (1)-(2) is free of directed cycles to begin with, or when (1)-(2) is a pure network flow problem, in which all cycles are linearly dependent.

For more general cases, first assume that the constraint matrix A, of (1)-(2) is Leontief. This assumption is easily dropped later. Each directed cycle of a Leontief directed hypergraph corresponds to a *generalized cycle matrix* which is a square matrix with a row for each vertex in the directed cycle and a column for each hyperarc in the directed cycle. In each column corresponding to hyperarc (J_i, v_{i+1}) there are two non-zero elements; $a + 1$ in the row corresponding to vertex v_{i+1} and $-a_{v_i}[J_i, v_{i+1}]$ in the row for vertex v_i.



$$A = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} & e_{12} & e_{13} \end{matrix} \\ \left[\begin{array}{ccccccccccccc} 1 & -1 & & & & & -1 & -1 & & & & & \\ & -1 & & 1 & & & & & & & & & \\ & 1 & -1 & & & & & & & & & & \\ & & 1 & -1 & & & & & & & & & -1 \\ & & & & -3 & & 1 & & & & & 1 & \\ & & & & 1 & -.9 & -1 & & & -2 & & & \\ & & & & & 1 & -.5 & & 1 & -4 & & & \\ & & & & -1 & & & 1 & & & & & 1 \\ & & & & & & & & & & 1 & -1 & -1 \end{array}\right] \end{array}$$

Fig. 1. Example hypergraph and constraint matrix.

Lemma 2.1. *Let C be a generalized cycle matrix. Then there is a solution to the system $Cx > 0$, $x \geq 0$ if and only if the directed cycle corresponding to C has gain > 1.*

Proof. If the gain is ≤ 1 we show there is no solution to the system $Cx > 0$, $x \geq 0$ by constructing a nonnegative vector $\lambda \neq 0$ such that $\lambda C \leq 0$. By a variant of Farkas only one of these systems can have a solution.

Assume without loss that the rows and columns of C have been permuted so that row i corresponds to vertex v_i and column i corresponds to hyperarc $e_i = (J_i, v_{i+1})$ in the directed cycle $v_1 e_1 v_2 e_2 v_3, \ldots, e_k v_{k+1}$ where $v_1 = v_{k+1}$. Assign a nonnegative set of multipliers to the rows of C by

$$\lambda_1 = 1, \qquad \lambda_i = 1 \Big/ \left(\prod_{j=i}^{k} a_{v_j}[J_j, v_{j+1}] \right), \quad i = 2, \ldots, k. \tag{4}$$

Then for $i=2,\ldots,k$ the ith component of λC is equal to zero since column i has a coefficient of $+1$ in row $i+1$ with multiplier $1/(\Pi_{j=i+1}^{k} a_{v_j}[J_j, v_{j+1}])$ and a coefficient of $-a_{v_i}[J_i, v_{i+1}]$ in row i with multiplier $1/(\Pi_{j=i}^{k} a_{v_j}[J_j, v_{j+1}])$. Also, by the hypothesis that the gain is $\leqslant 1$,

$$\begin{aligned}
&-a_{v_1}[J_1, v_2]+1\Big/\left(\prod_{j=2}^{k} a_{v_j}[J_j, v_{j+1}]\right)\\
&\quad=a_{v_1}[J_1, v_2]\left(-1+1\Big/\left(\prod_{j=1}^{k} a_{v_j}[J_j, v_{j+1}]\right)\right)\\
&\quad\leqslant 0,
\end{aligned}$$

so the first component of λC is nonpositive.

Conversely, if the cycle associated with generalized cycle C has gain >1, we construct solutions $x^i \geqslant 0$ such that $Cx^i \geqslant 0$ and strictly positive in the ith row. Summing these x^i gives $C(\sum_i x^i) > 0$.

It is sufficient to construct only x^1 since the other x^i take the same form after relabeling of the cycle. Assume vertex v_1 corresponds to the first row of C and consider

$$x^1_{e_k} = 1, \qquad x^1_{e_i} = \prod_{j=i+1}^{k} a_{v_j}[J_j, v_{j+1}], \quad i=1,\ldots,k-1, \tag{5}$$

again defining $v_{k+1} = v_1$.

With this solution there is a balance of flow at every vertex except v_1. At vertex v_1 the flow in is one unit since $x^1_{e_k} = 1$ and the flow out is the reciprocal of the cycle's gain. With gain >1, it follows that there is an excess flow into vertex v_1 and the first component of Cx^1 is strictly positive. □

Lemma 2.2. *If a Leontief substitution flow problem is gainfree and the associated constraint matrix A is Leontief, then the support hypergraph of every basic feasible solution is acyclic.*

Proof. Assume there is a basic feasible solution with a support hypergraph that contains a directed cycle. We show this directed cycle has a gain >1. Since A is Leontief, the basis matrix associated with the basic feasible solution for $b \geqslant 0$ is also Leontief (Veinott, 1968). Consider a Leontief basis B containing w.l.o.g. a directed cycle around vertices and hyperarcs in its upper left hand corner, and write it

$$B = \begin{bmatrix} C & R \\ S & D \end{bmatrix}.$$

That is, C is a square matrix with a row for each vertex in the directed cycle and a column for each hyperarc in the directed cycle. Since B is Leontief there is a nonnegative $\bar{x}$ with $B\bar{x} > 0$. Then $C\bar{x}^C + R\bar{x}^R > 0$. Since B is a square Leontief matrix

there is exactly one +1 in every row and in every column. Every column in the matrix C contains a +1 (corresponding to each hyperarc directed into each vertex in the directed cycle) so $R \leq 0$ which implies $C\bar{x}^C > 0$. Construct from C a generalized cycle matrix $\tilde{C}$ by eliminating the negative coefficients not along the cycle (that is, for every hyperarc $e_i = (J_i, v_{i+1})$ in the cycle delete all negative coefficients in C which do not correspond to vertex v_i). It follows that $\tilde{C}\bar{x}^C > 0$. Then by Lemma 2.1 the directed cycle has a gain >1. □

To extend Lemma 2.2 to all Leontief substitution flow problems assume A is pre-Leontief. Veinott (1968) shows the following two lemmas (as Theorems 2 and 3).

Lemma 2.3. *If A is pre-Leontief, then after permuting the rows and columns appropriately A can be written as*

$$\begin{bmatrix} A_1 & A_3 \\ 0 & A_2 \end{bmatrix}$$

where A_1 is Leontief and all the rows of A_2 are trivial. □

Partition $x = (x^1, x^2)$ to correspond to A_1 and A_2, respectively. Similarly for $b = (b^1, b^2)$. Since the rows of A_2 are trivial, $b^2 = 0$ in any feasible Leontief substitution flow.

Lemma 2.4. *Suppose A is pre-Leontief, A is partitioned as in Lemma 2.3, and $b^1 \geq 0$, $b^2 = 0$. Then the following are equivalent:*

(i) *$x = (x^1, x^2)$ is an extreme point of the Leontief flow problem.*

(ii) *x^1 is an extreme point of the Leontief flow problem defined by constraint matrix A_1 and right hand side b^1, and $x^2 = 0$.* □

Theorem 2.5. *If a Leontief substitution flow problem is gainfree, then the support hypergraph of every basic feasible solution is acyclic.*

Proof. Let $x = (x^1, x^2)$ be a basic feasible solution to a Leontief flow problem. Since the solution is feasible, $b^1 > 0$ and $b^2 = 0$. Then by Lemma 2.4, $x^2 = 0$ and x^1 is an extreme point a Leontief flow problem with a Leontief constraint matrix (i.e. A_1). By Lemma 2.2 the support hypergraph of the Leontief flow problem corresponding to A_1 and b^1 is acyclic. The result now follows since $x^2 = 0$. □

The support hypergraph of a basic feasible solution actually has a much stronger property than simply being acyclic.

Theorem 2.6. *If a Leontief substitution flow problem is gainfree then the support hypergraph of every basic feasible solution is a hyperforest and each component of the hyperforest is a hyperarboresence.*

Proof. Following Lemma 2.4 and Theorem 2.5, we know that if $\bar{x} = (\bar{x}^1, \bar{x}^2)$ is an extreme point of the Leontief flow problem, then $\bar{x}^1$ is an extreme point of the

Leontief flow problem defined by $A_1 x = b_1$ where A_1 is Leontief. The hyperarcs with positive flow are components of $\bar{x}^1$. Since A_1 is Leontief, the basis matrix which defines $\bar{x}^1$ is also Leontief. This Leontief basis matrix has exactly one $+1$ in each row and each column. Then at most one hyperarc with positive flow is directed into a vertex. Since the support is acyclic by Lemma 2.2 and there is at most one hyperarc directed into any vertex, it must be a hyperforest where each component is a hyperarboresence. □

Corollary 2.7. *If a Leontief substitution flow problem with a Leontief constraint matrix is gainfree, then the basis matrix of every basic feasible solution can be permuted into a lower triangular matrix.*

Proof. Any basic feasible solution for a given nonnegative right hand side is a basic feasible solution for all nonnegative right hand sides, so we assume without loss that $b_i > 0$ for all i. Thus, in the basic solution there is a nonzero hyperarc directed into every vertex so the solution is not degenerate.

By Theorem 2.6 the corresponding support hypergraph is a hyperforest where each component is a hyperarboresence. Each hyperarboresence (since it is acyclic) induces an ordering on the vertices such that the head vertex of every hyperarc can be assigned a unique number which is larger than the unique number assigned to all of the tail vertices. This is true for the entire hyperforest since it is true for every component.

Use this assignment to permute the $|V| \times |V|$ constant matrix corresponding to hyperarcs with positive flow as follows. Order the rows of the matrix by descending vertex number. By definition of hyperarboresence, there is exactly one nonzero basic hyperarc directed into each vertex. Thus, the columns can be ordered such that the column corresponding to hyperarc (J_1, v_1) follows the column corresponding to hyperarc (J_2, v_2) if and only if vertex v_1 is assigned a lower number than vertex v_2. With this permutation each diagonal element is a plus one and the negative elements corresponding to the tail weights lie below the diagonal since the tail vertices were assigned lower numbers than the head. □

Lemma 2.8. *Given a Leontief directed hypergraph $G = (V, H)$:*

(i) *If hyperarc (J, i) is nondegenerate in any Leontief substitution flow problem on G, then every vertex in J is nontrivial.*

(ii) *If every vertex $j \in J$ is nontrivial for hyperarc (J, i) then vertex i is nontrivial.*

Proof. (i) By definition, if hyperarc (J, i) is nondegenerate then there is an extreme point solution with $x[J, i] > 0$. Then, by Lemma 2.4, hyperarc (J, i) corresponds to a column in matrix A_1. By definition of A_1 every vertex in J corresponds to a row in A_1 and therefore is nontrivial.

(ii) If all $j \in J$ are nontrivial, then for each j there is a feasible solution $\bar{x}^j \geq 0$ to some Leontief flow problem with right hand side $b^j \geq 0$ and $b^j_j > 0$. It follows that

$\sum_{j\in J} \bar{x}^j$, together with a flow on $x[J, i]$ of $\phi = \min\{b^j_j / a_j[J, i]\}$, is feasible for some $b^i \geq 0$ with $b^i_i = \phi > 0$. □

In terms of the Leontief flow paradigm, the columns of A_1 in Lemma 2.4 correspond to all of the hyperarcs (J, i) with the property that vertex i and all $j \in J$ are nontrivial. However, it is possible to construct example Leontief directed hypergraphs which contain degenerate hyperarcs (J, i) with every vertex $j \in J$ nontrivial. That is, the converse of Lemma 2.8(i) is not true, and there may be columns of A_1 which correspond to degenerate hyperarcs. Part (ii) of Lemma 2.8 says that if there is a solution with net positive flow into all $j \in J$, then existence of hyperarc (J, i) implies there is a solution with net positive flow into vertex i.

3. Strongly polynomial algorithms

Assign dual variables u_i to constraints i of (1). Then the dual feasibility constraints for a Leontief flow problem on Leontief directed hypergraph $G = (V, H)$ are

$$u_i \leq c[J, i] + \sum_{j\in J} a_j[J, i]u_j \quad \text{for all } (J, i) \in H, \tag{6}$$

$$0 \leq c[J, \emptyset] + \sum_{j\in J} a_j[J, \emptyset]u_j \quad \text{for all } (J, \emptyset) \in H. \tag{7}$$

As usual in linear programming theory, a solution u is said to be *complementary* with respect to a primal flow x if (J, i) in the support of x implies the corresponding inequality of (6) or (7) holds as an equality.

Algorithmic research in Leontief systems has centered on finding solutions to (6)-(7) by successive approximation schemes based on (6). Finite convergence is rare. Numerous procedures and asymptotic convergence results are available. See Koehler et al. (1975) and Veinott (1969b).

Our focus here is on subclasses of Leontief flow problems admitting provable finite and polynomially bounded solution times via successive approximation. The simplest such successive approximation procedure corresponds to Jacobi iteration with an initial starting value of big M and extends the classic Bellman-Ford method for the shortest path problem as follows:

Value Iteration Algorithm.

Step 1. Initialize: $u^0_i \leftarrow M$, $nontriv(i) =$ "true" for all $(\emptyset, i) \in V$, otherwise $nontriv(i) =$ "false"; $t[i] \leftarrow 0$ for all $i \in V$; $t \leftarrow 0$.

Step 2. Iterative step: while some u^t_i changed; $t \leftarrow t+1$ and for all vertices i with nonempty $\{(J, i) \in H\}$ do:

$$c[\bar{J}, i] + \sum_{j\in\bar{J}} a_j[\bar{J}, i]u^{t-1}_j = \min\left\{ c[J, i] + \sum_{j\in J} a_j[J, i]u^{t-1}_j \;\middle|\; (J, i) \in H \right\}, \tag{8}$$

$$u^t_i \leftarrow \min\left\{ u^{t-1}_i, c[\bar{J}, i] + \sum_{j\in\bar{J}} a_j[\bar{J}, i]u^{t-1}_j \right\}. \tag{9}$$

If $u_i^{t-1} > c[\bar{J}, i] + \sum_{j \in \bar{J}} a_j[\bar{J}, i] u_j^{t-1}$ and $nontriv(j) =$ "true" for all $j \in \bar{J}$,

$$\text{record } J[i] \leftarrow \bar{J},\ t[i] \leftarrow t,\ nontriv(i) \leftarrow \text{"true"}. \tag{10}$$

If $\{(J, i) \in H\} = \emptyset$, $i \in V$ then $u_i^t = u_i^{t-1}$.

In the Value Iteration Algorithm, $nontriv(i)$ is a flag which is set to "true" when the algorithm discovers that vertex i is nontrivial. It is based on the fact that ⟨1⟩ all vertices which have a tailless hyperarc $(\emptyset, i)$ are nontrivial, and ⟨2⟩ if all the vertices in J are nontrivial, then the existence of hyperarc (J, i) implies vertex i is nontrivial by Lemma 2.8. We can take big M in the initialization step to be any real number larger than $2^{(1+4|H|^2\phi)}$ where ϕ is the size of the largest inequality in (6). See Schrijver (1986, p. 121).

Lemma 3.1. *If $\bar{u}_i \leq M$, $i \in V$ is a feasible solution to the dual constraints (6) for Leontief directed hypergraph $G = (V, H)$ and criterion vector c, then $u^t \geq \bar{u}$ for all t where u^t is calculated by Value Iteration.*

Proof. By induction on t.

If $t = 0$, $u_i^0 = M \geq \bar{u}_i$, all $i \in V$. Assume $u_i^t \geq \bar{u}_i$, all $i \in V$. Then either $u_i^{t+1} = u_i^t$, or

$$u_i^{t+1} = \min\left\{ c[J, i] + \sum_{j \in J} a_j[J, i] u_j^t \,\middle|\, (J, i) \in H \right\}$$

$$\geq \min\left\{ c[J, i] + \sum_{j \in J} a_j[J, i] \bar{u}_j \,\middle|\, (J, i) \in H \right\}$$

$$\geq \bar{u}_i \quad \text{(by (6))}. \qquad \square$$

The key result of Section 2, Theorem 2.5, is that basic feasible solutions to gainfree Leontief substitution flow problems have acyclic support. Whether or not caused by the gainfree property, an acyclic basic feasible solution with some support hyperarc pointing into every nontrivial vertex assigns implicit *levels* to the nontrivial vertices. Each vertex i serviced by a source hyperarc $(\emptyset, i)$ may be taken as level 1, and the level of other nontrivial i in the support is defined recursively as $1 +$ max{levels of $j \in J \,|\, (J, i) \in H$}.

Lemma 3.2. *Consider an m-vertex Leontief flow problem on Leontief directed hypergraph $G = (V, H)$. If there exists a basic feasible flow $\bar{x}$ having acyclic support with a hyperarc directed into every nontrivial vertex, and a complementary dual solution $\bar{u}_i \leq M$ feasible in (6), then for all nontrivial $i \in V$, $u_i^t = \bar{u}_i$ after at most $t = m$ iterations.*

Proof. By hypothesis the support hypergraph is acyclic and has a hyperarc directed into each nontrivial i, which means that each nontrivial i can be assigned a level as described above. Then feasibility of $\bar{u}$ in (6) and complementary slackness imply for nontrivial i,

$$\bar{u}_i = \min\left\{ c[J, i] + \sum_{j \in J} a_j[J, i]\bar{u}_j \;\middle|\; (J, i) \in H \right\}. \tag{11}$$

Proceed by induction on the level l, induced by $\bar{x}$, on the nontrivial vertices i. For $l = 1$, the first iteration of Value Iteration sets $u_i^1 = c[\emptyset, i]$ and complementary slackness gives $\bar{u}_i = c[\emptyset, i]$. Then by Lemma 3.1 and the monotonicity of Value Iteration,

$$c[\emptyset, i] = u_i^1 \geqslant u_i^t \geqslant \bar{u}_i = c[\emptyset, i]$$

so that u_i^t is unchanged after iteration $l = 1$.

Inductive step. Assume now that for every nontrivial vertex i of level $l \leqslant \sigma$, $u_i^t = \bar{u}_i$ for $t \geqslant l$. If vertex i is level $l = \sigma + 1$, and (J, i) is a hyperarc directed into i, then for every $j \in J$ the level of j is at most l. Then at iteration $l+1$,

$$\begin{aligned} u_i^{l+1} &\leqslant \min\left\{ c[J, i] + \sum_{j \in J} a_j[J, i]u_j^l \;\middle|\; (J, i) \in H \right\} \\ &= \min\left\{ c[J, i] + \sum_{j \in J} a_j[J, i]\bar{u}_j \;\middle|\; (J, i) \in H \right\} \\ &= \bar{u}_i. \end{aligned}$$

The first equality is by the induction hypothesis, the second by (11). Again by Lemma 3.1 and the monotonicity of Value Iteration, $\bar{u}_i \leqslant u_i^{l+1}$, and this implies $u_i^t = \bar{u}_i$ for all $t \geqslant l+1$. To complete the proof observe that an acyclic hypergraph on $m = |V|$ vertices can have vertices of level no greater than m. □

Informally, Lemma 3.2 says that for any Leontief flow problem having a basic feasible solution with acyclic support hypergraph containing a hyperarc directed into all nontrivial i, then any complementary dual solution is finitely computed by Value Iteration.

Value Iteration is a purely dual algorithm. If it stops, however, it is easy to retrieve a corresponding primal flow for any demand vector $\tilde{b} \geqslant 0$ that is zero at trivial vertices. We use the labels $J[i]$ and $t[i]$ saved at Step 2 as follows:

Primal Retrieval.

Step 1. Initialization: set $\tilde{x}[J, i] \leftarrow 0$ for all $(J, i) \in H$, create active vertex list $\tilde{V} \leftarrow \{i \mid i \text{ nontrivial}\}$, and establish vertex flows f_i as $f_i \leftarrow \tilde{b}_i$ for all $i \in \tilde{V}$.

Step 2. Iterative step: while $\hat{V} \neq \emptyset$, choose an $i \in \tilde{V}$ with maximum $t[i]$, pick the inbound hyperarc $(J[i], i)$ from (10) and update

$$\tilde{x}[J[i], i] \leftarrow f_i,$$

$$f_j \leftarrow f_j + a_j[J[i], i]\tilde{x}[J[i], i] \quad \text{for all } j \in J[i],$$

$$\tilde{V} \leftarrow \tilde{V} \backslash \{i\}.$$

To establish correctness of Primal Retrieval we first prove two lemmas.

Lemma 3.3. *If Value Iteration terminates finitely, then for all vertices i with nontriv(i) =* "true", *labels J[i] and t[i] satisfy*

$$u_i^{t[i]} = c[J[i], i] + \sum_{j \in J[i]} a_j[J[i], i] u_j^{t[i]-1}, \tag{12}$$

$$t[j] \leqslant t[i] - 1 \quad \textit{for all } j \in J[i]. \tag{13}$$

Proof. For each i with *nontriv*(i) = "true", labels $J[i]$ are defined, and (12) follows from (9). For (13) assume there is an i and a $j' \in J[i]$ with $t[j'] > t[i] - 1$. It follows from (9)–(10) that $u_{j'}^{t[i]-1} > u_{j'}^{t[j']}$. Then, since all $a_j[J[i], i]$ are positive,

$$u_i^{t[i]} > c[J[i], i] + \sum_{j \in J[i]} a_j[J[i], i] u_j^{t[j']}$$

and u_i will change after its last change $t[i]$. □

Lemma 3.4. *Consider an m-vertex Leontief flow problem on Leontief directed hypergraph* $G = (V, H)$.

(i) *If vertex i is trivial, then nontriv*(i) = "false" *at every step t of Value Iteration.*

(ii) *Assume there exists a basic feasible flow* $\bar{x}$ *on G with an acyclic support having a hyperarc directed into all nontrivial vertices i. If either Value Iteration converges after* $j \leqslant m+1$ *iterations leaving nontriv*(i) = "false", *or nontriv*(i) *remains* "false" *after* $m+1$ *iterations, then vertex i is a trivial vertex.*

Proof. First prove (i) by contrapositive and show *nontriv*(i) = "true" for some t implies vertex i is nontrivial. If *nontriv*(i) = "true" there is a first iteration $s(i)$ such that *nontriv*(i) = "true". Proceed by induction on $s(i)$. If $s(i) = 1$ then there exists hyperarc $[\emptyset, i]$ and i cannot be trivial.

Inductive step. Assume that all vertices j with $s(j) \leqslant \sigma$ are nontrivial. If $t = s(i) = \sigma + 1$ then by (10) there is a hyperarc $[J[i], i]$ directed into i such that all $j \in J[i]$ had *nontriv*(j) marked "true" at an earlier iteration. By the induction hypothesis, all such $j \in J[i]$ are nontrivial. Then by Lemma 2.8 vertex i is nontrivial.

Now prove part (ii) by contrapositive and assume vertex i is nontrivial and show for some iteration $t \leqslant m$ of Value Iteration that *nontriv*(i) = "true". By hypothesis there is a basic feasible flow $\bar{x}$ with an acyclic support and at least one hyperarc directed into every nontrivial vertex. Thus, every nontrivial vertex can be assigned

a level l. By using an inductive argument identical to the one used in Lemma 3.2 it follows that if nontrivial vertex has level l then $nlntriv(i) =$ "true". The result follows from the fact that for an m-vertex Leontief flow problem $l \leq m$. □

Lemma 3.5. *Consider an m-vertex Leontief flow problem on Leontief directed hypergraph $G = (V, H)$ with $\tilde{b}_i \geq 0$ for all i, and $\tilde{b}_i = 0$ for all trivial i. If Value Iteration is finite and $nontriv(i) =$ "true" for all vertices i with $\tilde{b}_i > 0$, then Primal Retrieval computes a flow $\tilde{x}$ that is feasible for $\tilde{b}$, complementary with the u^t obtained when Value Iteration stopped, and having a support which is a hyperforest, each component of which is a hyperarboresence.*

Proof. Since the cardinality of $\tilde{V}$ strictly decreases at each Step 2 in Primal Retrieval, the algorithm is finite; Step 2 can be executed at most m times. We next show the Primal Retrieval algorithm is well defined given the hypotheses. By part (ii) of Lemma 3.4, $nontriv(i) =$ "true" for all nontrivial vertices i, which, by (10), implies $u_i^{t[i]} < u_i^{t[i]-1}$. Again by (10), this guarantees the existence of a $J[i]$ for each nontrivial i. Furthermore, when Value Iteration Step 2 assigns $J[i]$, it does so on the basis of a computation with $nontriv(i) =$ "true" for every $j \in J[i]$. Then by part (i) of Lemma 3.4 every vertex in $J[i]$ is nontrivial. Hence only nontrivial vertices are assigned flow in Primal Step 2. For the selected $t[i], j \in J[i]$ implies $t[j] < t[i] = \max\{t[k] \mid k \in \tilde{V}\}$. Then all $j \in J[i]$ are selected at a future execution of Primal Step 2 and conservation of flow at these vertices is maintained. Thus, when Primal Retrieval stops, $\tilde{x}$ is feasible for $\tilde{b}$. Nonnegativity of $\tilde{b}$ implies nonnegativity of $\tilde{x}$.

To see that complementary slackness is maintained, note label $J[i]$ is last changed at vertex i at iteration $t[i]$ when u_i is last changed. Then by (9)-(10) constraint (6) for hyperarc $(J[i], i)$ is satisfied as an equality. Since Primal Retrieval assigns flow only to such hyperarcs it follows that $\tilde{x}$ is complementary with the u^t. Note that Primal Retrieval never constructs a solution with positive flow on sink hyperarcs $(J, \emptyset)$ so the solution is always complementary to (7).

Finally, by construction at most one inbound hyperarc at any vertex is assigned positive flow. Again, because the selected i do not repeat, it follows that the support hypergraph does not contain any directed cycles, i.e. it is a hyperforest. Since there is at most arc directed into each vertex, each component of the hyperforest is a hyperarboresence. □

We have stated Lemmas 3.2-3.4 to encompass *all* Leontief flow problems having a basic flow with acyclic support with a hyperarc directed into nontrivial vertex. Indeed, only Lemma 3.5 requires Leontief substitution assumption $b \geq 0$ in order that Primal Retrieval give a nonnegative solution. We are now ready to apply these results to see that Value Iteration and Primal Retrieval resolve every possible $b \geq 0$, gainfree input in strongly polynomial time.

Theorem 3.6. *If Value Iteration is applied to a gainfree Leontief substitution flow problem with m vertices then*:

(i) *The Leontief flow problem has an optimal solution if and only if Value Iteration terminates after* $j < m+1$ *iterations with* u^j *satisfying* (6)-(7), *and* $nontriv(i) =$ "true" *whenever* $b_i > 0$.

(ii) *The Leontief flow problem is infeasible if and only if after* $m+1$ *iterations* $nontriv(i) =$ "false" *for some i with* $b_i > 0$, *or the Value Iteration algorithm terminates after* $j \leq m+1$ *iterations with* $nontriv(i) =$ "false" *for some i with* $b_i > 0$.

(iii) *The Leontief flow problem is unbounded if and only if* $u_l^{m+1} \neq u_l^m$ *for some l and* $nontriv(i) =$ "true" *for all i with* $b_i > 0$; *or, Value Iteration terminates after* $j \leq m+1$ *iterations with* $c[J, \emptyset] + \sum_{l \in J} a_l[J, i] u_l^j < 0$ *for some* $(J, \emptyset) \in H$ *and* $nontriv(i) =$ "true" *for all i with* $b_i > 0$.

Proof. (i) Assume for some $j \leq m+1$, $u^j = u^{j-1}$, $nontriv(i) =$ "true" if $b_i > 0$, and u^j satisfies (6)-(7). Since $nontriv(i) =$ "true" whenever $b_i > 0$, it follows from part (i) of Lemma 3.4 that vertex i is nontrivial. Then by Lemma 3.5 there exists a primal feasible flow $\tilde{x}$ complementary to u^j. Since (6) and (7) are satisfied by u^j, we have dual feasibility which implies $\tilde{x}$ is primal optimal.

Now assume the Leontief flow problem for a given $\bar{b} \geq 0$ has an optimal solution and thus an optimal basic solution $\bar{x}$. Then there exists a corresponding optimal dual solution $\bar{u}$ with components less than M. Using the notation developed for Lemma 2.3, let $\bar{u}^1$ be the optimal set of dual variables for the non-trivial constraints, i.e. the constraints in the system $A_1 x^1 = \bar{b}^1$. Since A_1 is Leontief every matrix B_1 contained in A_1 corresponding to a basic feasible solution is also Leontief. The inverse of a square Leontief matrix is nonnegative. Then $(B_1)^{-1} b^1 \geq 0$ for all nonnegative b^1 and without loss we assume $\bar{b}^1 > 0$. By hypothesis the Leontief flow problem is gainfree so by Theorem 2.5, the support corresponding to $(B_1)^{-1} \bar{b}^1$ with $\bar{b}^1$ positive is acyclic and complementary to $\bar{u}^1$. The support is also complementary to the full $\bar{u}$ since there is no flow into trivial vertices. Since $\bar{b}^1 > 0$ there is a hyperarc in the support directed into every nontrivial vertex. Then by Lemma 3.2 after at most $k \leq m+1$ iterations of Value Iteration, $u_i^k = \bar{u}_i$ when i is nontrivial. If i is trivial, the u_i^l calculated by Value Iteration are exactly the same as if an artificial hyperarc $(\emptyset, i)$ with cost M had been added to the problem. Since there is an optimal solution to this problem, applying Simplex to this problem with the artificial arcs will yield a basic feasible solution having acyclic support (which will now include the artificial arcs on the trivial vertices) and an optimal complementary $\bar{u}$. Since these artificial hyperarcs have no effect on the dual variables calculated by Value Iteration, using Lemma 3.2 again implies that Value Iteration will terminate after at most $k \leq m+1$ iterations with $u_i^k = \bar{u}_i$.

(ii) If the Leontief flow problem is infeasible then $b_i > 0$ for some trivial vertex i. If i is a trivial vertex, then by part (i) of Lemma 3.4 $nontriv(i) =$ "false" for every iteration t. Now assume that $b_i > 0$ for some i and that after $m+1$ iterations $nontriv(i) =$ "false", or the Value Iteration algorithm terminates after $j \leq m+1$

iterations with $nontriv(i) =$ "false". Since the hypergraph is gainfree there exists a basic feasible solution with acyclic support for all $b_k > 0$, k nontrivial. Then it follows from part (ii) of Lemma 3.4 that vertex i is trivial. The problem must be infeasible.

(iii) If a Leontief flow problem is unbounded then Value Iteration either terminates within $m+1$ iterations, or it does not. If Value Iteration does not terminate after $m+1$ iterations, then $u_l^{m+1} \neq u_l^m$ for some l, yet $nontriv(i) =$ "true" for all i with $b_i > 0$ or the problem would be infeasible by part (ii). If Value Iteration terminates after $j \leq m+1$ iterations then by part (ii) of Lemma 3.4 if $nontriv(i) =$ "false", vertex i is trivial. If $b_i > 0$ the problem would be infeasible, thus $b_i = 0$ because the problem is unbounded. Since Value Iteration has terminated the u^j satisfy (6). Then by part (i), since the problem is unbounded, u^j cannot satisfy (7).

Conversely, if $u_l^{m+1} \neq u_l^m$ for some l and $nontriv(i) =$ "true" for all i with $b_i > 0$; or, the Value Iteration algorithm terminates after $j \leq m+1$ iterations with $c[J, \emptyset] + \sum_{l \in J} a_l[J, i] u_l^j < 0$ for some $(J, \emptyset) \in H$ and $nontriv(l) =$ "true" for all l with $b_l > 0$ then by (i) and (ii) the problem cannot be infeasible and cannot have an optimal solution. Therefore the problem is unbounded. □

Corollary 3.7. *If a gainfree Leontief substitution flow problem on m vertices has an optimal solution, the Value Iteration Algorithm converges to an optimal dual solution on* $O(mp)$ *time, and Primal Retrieval constructs an optimal primal solution in* $O(p)$ *time.*

Proof. By Theorem 3.6, part (i) the Value Iteration algorithm converges to an optimal dual solution in $m+1$ or fewer iterations. The work of Step 2 given by (8)-(10) requires at most $O(p)$ computations.

For the Primal, maintain linked lists of all $i \in \tilde{V}$ with $t[i] = k$, $k \in \{1, \ldots, m\}$. These lists can be constructed in $O(m) \leq O(p)$ time and at each Step 2 execution we flag the nonempty list with the largest $t[i]$. Then the m executions of Step 2 do a total of $O(m) \leq O(p)$ extractions and $\hat{x}$ settings, plus f_j updates. Initialization for vertices is $O(m)$ and for edges $O(n) \leq O(p)$. Total effort is $O(p)$. □

Theorem 3.6 can also be used to give a strongly polynomial algorithm for testing if a Leontief directed hypergraph is gainfree. To test gainfreeness of a given $G = (V, H)$, construct an associated ordinary directed graph $G' = (V, H')$ on the same vertex set. Arc set H' contains an arc (j, i) for each $j \in J$ of each hyperarc $(J, i) \in H$ that has both tail(s) and a head. The cost on such (j, i) is $\log(a_j[J, i])$, and all net demands $b_i = 0$.

By construction, G' is gainfree, since all tail weights are 1, and the associated Leontief substitution flow problem is feasible because $b = 0$. Thus Theorem 3.6 applies, and Value Iteration will settle in strongly polynomial time whether the problem for G' is optimal or unbounded. Well known theory for ordinary network flows establishes that the problem can be unbounded if and only if there is a directed

cycle of negative total length. Thus the problem is unbounded if and only if some directed cycle in the original G has, in the notation of (3), $0 > \sum_{i=1}^{k} \log(a_{v_i}[J, v_{i+1}]) = \log(\prod_{i=1}^{k} a_{v_i}[J, v_{i+1}])$, which implies gain is >1.

We have included a full treatment of Value Iteration polynomiality in this section to make our directed hypergraph development of Leontief substitution flow problems complete. However, much of the work of this section appears elsewhere in other forms. Erickson (1978, 1988) proved essentially the same result as Lemma 3.2 for the case where A is Leontief. Ullman and Van Gelder (1988) also gave for a finite proof of Value Iteration (their Algorithm 3) when the A matrix is integral, and hence gainfree. Readers may also wish to consult the paper by Adler and Cosares (1989) where a strongly polynomial algorithm is given for any Leontief substitution problem with at most two nonzero elements in each column. Finally, Charnes and Raike (1966) give a strongly polynomial algorithm for the generalized network problem with nonnegative b. Their algorithm requires either an acyclic graph, or nonnegative costs and tail weights ≥ 1; hence the problem is gainfree.

4. Integral and binary solutions

Much of the recent interest in Leontief flow problems has resulted from the fact that the Value Iteration and Primal Retrieval algorithms often yield integral, and even binary integral solutions.

Jeroslow and Wang (1989) show, in a logic context, that (in the present terminology) for Leontief substitution flow problems over integral A and b, basic feasible solutions correspond to acyclic support hypergraphs. It follows immediately that optimal solutions are integral (see, e.g. Martin, Rardin and Campbell, 1990). Noting that a Leontief flow problem with an integral coefficient matrix is gainfree under definition (3), we extend these results as consequences of theorems in the previous two sections.

Given integral A and rational b the linear program $\min\{cx \mid Ax = b, x \geq 0\}$ is totally dual integral (TDI) if the dual problem has an integral optimal solution for every integral vector c for which it has an optimal solution. See Edmonds and Giles (1977) or Giles and Pulleyblank (1979). Total dual integrality is weaker than the more familiar total unimodularity, but is still sufficient to guarantee integral primal optima when b is an integral vector.

Theorem 4.1. *Every Leontief substitution flow problem with an integral coefficient matrix A and rational right hand side b is TDI.*

Proof. If A is integral then by (3) the Leontief flow problem is gainfree. Then by Theorem 3.6 Value Iteration will compute an optimal dual solution in m iterations. If A is integral then the $a_j[J, k]$ of update (8)-(9) are integral, so given integral costs $c[J, k]$ the algorithm only assigns integers to u_i^l. Thus, for any integral c

yielding a finite optimum, the optimal dual solution constructed by Value Iteration is integral. □

The following theorem is an immediate consequence of Theorem 4.1.

Theorem 4.2. *If a Leontief substitution flow problem with integral A, b has an optimal solution, then it has an integral optimal solution.* □

Theorem 4.2 for A Leontief is also easily derived from our Corollary 2.7 and Theorem 7, of Veinott (1968). Veinott shows that the condition B^{-1} integral for all Leontief basis matrices is equivalent to the condition that for all feasible, nonnegative integral b, all the extreme points of the feasible region are integral. Our Corollary 2.7 implies basis inverses have the needed property.

Theorem 4.3. *Every feasible basis submatrix B of a Leontief substitution flow problem with integral, Leontief A has an integral inverse B^{-1} that can be permuted into a lower triangular matrix.*

Proof. If A is integer and Leontief, the corresponding Leontief directed hypergraph is gainfree, and Corollary 2.7 applies. The proof of that corollary shows that every feasible basis submatrix B can be permuted until B is lower triangular with $+1$'s on the diagonal. Inversion by say Gaussian elimination easily establishes that the corresponding basis inverse is integral and lower triangular. □

Although integrality results are important in applications such as Horn clause knowledge bases (see Section 5.2), binary integrality results are required for applications involving polyhedral combinatorics and mixed 0/1 linear programming. Figure 2 illustrates that Leontief substitution flow problems need not possess this binary integrality property. In fact, given integral data, any basic solution with flow on a hyperarc which has at least one tail weight >1 will not be binary since the support of the basic solution is a hyperforest and each component is a hyperarboresence. Similarly, a $b_i > 1$ gives basic solutions which are not binary. Even when b is binary, and all coefficient in A are $0, \pm 1$, basic solutions are not necessarily binary. This is illustrated in Figure 2, where the unique feasible flow is $x_{e_2} = x_{e_3} = x_{e_4} = 1$, $x_{e_1} = 2$.

In Figure 2 the problem is caused by the presence of the paracycle. In this section we show that for binary b, and A a matrix with $0, \pm 1$ elements, the absence of paracycles in supports corresponding to basic feasible solutions gives a sufficient and necessary condition for binary integrality. Before proceeding, we note that there is no loss of generality in assuming Leontief flow problems with binary b have unit vector right hand sides. If the given problem has a binary b which is not a unit vector, we may construct an equivalent Leontief flow model by adding a new vertex v and an additional hyperarc (J, v) with $J = \{i \mid b_i = 1\}$. The demand vector for the revised problem has $b_i = 1$ for vertex $i = v$ only. (Note, however, that this reformulation can create new paracycles.) We can also establish that subhyperarboresences in a support hypergraph will persist if $b \geqslant 0$ are restricted to unit vectors.

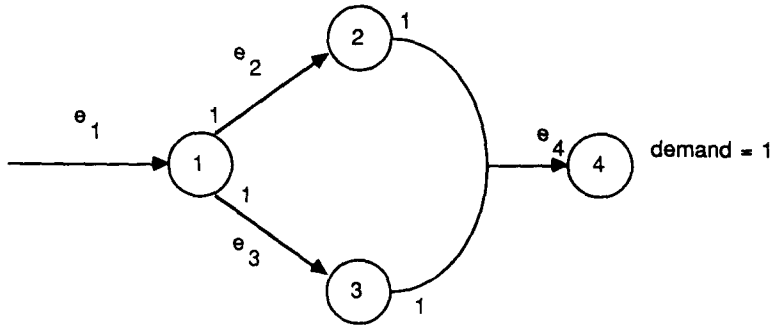


Fig. 2. Leontief flow problems with nonbinary optima.

Lemma 4.4. *Let F be the support hypergraph of a basic feasible solution to Leontief flow problem on gainfree Leontief directed hypergraph $G=(V, H)$ with right hand side $b \geq 0$, and vertex k demand $b_k > 0$. Then the subhyperarborescence induced by all vertices of V on directed paths leading to k in F is the support hypergraph of a basic feasible solution to the Leontief flow problem on the same G with unit vector right hand side e^k.*

Proof. The basic feasible solution of which F is the support must be the unique optimal solution for some costs $\{c[J, i]\}$. Since G is gainfree and $b \geq 0$, we know by Theorem 3.6 that Value Iteration over such costs will terminate finitely, and by Lemma 3.5 that Primal Retrieval will assign optimal flows positive on F. Furthermore, since Value Iteration does not depend on b, Primal Retrieval will construct the unique optimal solution for any nonnegative right hand side that is 0 on trivial vertices. Noting there is a feasible solution with $b_k > 0$, vertex k is nontrivial, and Primal Retrieval will construct the support for unit vector right hand side e^k using exactly the same labels $t[i]$ and $J[i]$ it employed to generate flows for b.

Under Theorem 2.6, F is a hyperforest where each component is a hyperarborescence, so that the component induced by vertices with a path to k must consist of exactly those $(J[i], i)$ encountered as Primal Retrieval proceeds backward from k. With the right hand side at k positive in both the b and the e^k cases, flows on all such hyperarcs will be positive in both solutions. Thus the support for e^k will be exactly the induced subgraph in F. □

Define sets

$$I_k[l] \stackrel{\text{def}}{=} \{s \mid \text{there is a source hyperarc } (\emptyset, s) \text{ directed into } s, \text{ and there is a directed path of nondegenerate hyperarcs from } s \text{ to } l \text{ which does not contain vertex } k\}. \tag{14}$$

A Leontief directed hypergraph is *disjointly reachable* if

$$I_k[j_1] \cap I_k[j_2] = \emptyset \quad \text{for every nondegenerate } (J, k) \in H, \; j_1, j_2 \in J, \; j_1 \neq j_2. \tag{15}$$

That is, a Leontief directed hypergraph is disjointly reachable if for every hyperarc (J, k) and source vertex v_0, the existence of a directed path of nondegenerate hyperarcs from v_0 to $j_1 \in J$, which does not contain vertex k, implies every directed path of nondegenerate hyperarcs from v_0 to $j_2 \in J$ must contain vertex k. The Leontief directed hypergraph $G = (V, H)$ is *support disjointly reachable* if for all $b \geq 0$, the support hypergraph of every basic feasible solution to the corresponding Leontief substitution flow problem is disjointly reachable. See Figure 3.

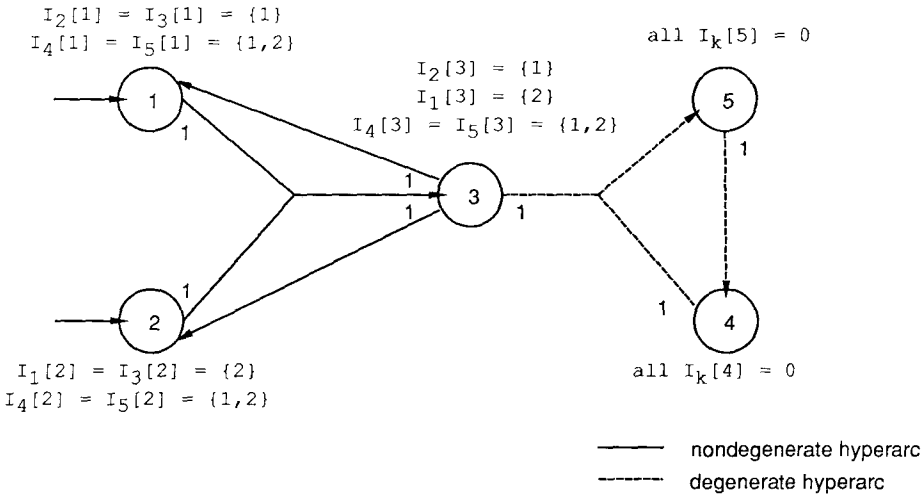


Fig. 3. Example of a disjointly reachable hypergraph.

Theorem 4.5. *The following are equivalent for Leontief flow problems on Leontief directed hypergraph $G = (V, H)$ with constraint matrix A consisting of* $0, \pm 1$:

(i) *The inverse of every basis submatrix feasible for some $b \geq 0$ is binary in all columns i for which $b_i > 0$.*

(ii) *Extreme flows are binary for every unit vector right hand side $b = e^i$ for which there is a feasible flow.*

(iii) *No support hypergraph corresponding to a basic feasible solution for some $b \geq 0$ contains a paracycle.*

(iv) *G is support disjointly reachable.*

Proof. (i)$\Rightarrow$(ii) Let B be any basis matrix such that $B^{-1}e^i \geq 0$. By hypothesis B^{-1} is binary in column i. Thus, the extreme point solution $B^{-1}e^i$, which equals column i of B^{-1}, is also binary.

(ii)$\Rightarrow$(iii) Prove the contrapositive. Assume that a given Leontief directed hypergraph with coefficients of $0, \pm 1$ in the constraint matrix has a paracycle P contained in the support hypergraph $\bar{P}$ corresponding to a basic feasible solution for $b \geq 0$.

Let the paths defining the paracycle begin at vertex r, and merge in hyperarc (J, k). Now observe that the unique hyperarc (J, k), directed into k has a positive flow, so that Lemma 4.4 assures the paracycle will persist in the support for unit vector right hand side e^k. In this new support, $x[J, k] = 1$. But then the two hyperarcs preceding (J, k) in the paths defining P must also have a flow of one unit since $\bar{P}$ which contains P is a hyperarboresence. Continuing in this way, we conclude that at least 2 units of flow must leave the root vertex r of P. It follows that the unique hyperarc of $\bar{P}$ directed into r has flow ≥ 2 and hence the extreme flow is not binary for right hand side e^k.

(iii)$\Rightarrow$(iv) Again, prove the contrapositive. If G is not support disjointly reachable, then there exists a $b \geq 0$ for which the support hypergraph of the corresponding Leontief flow problem contains a hyperarc (J, k), distinct tail vertices $j_1, j_2 \in J$, and directed paths P_1 and P_2 leading from some source vertex s to j_1 and j_2, respectively, without passing through vertex k and using ly nondegenerate hyperarcs (since all hyperarcs in the support are nondegenerate). Paths P_1 and P_2 have at least one common vertex s. Let r be the last vertex they have in common before they reach their respective tails j_1 and j_2. Portions of P_1 and P_2 beginning with r, together with hyperarc (J, k), form a paracycle in the support.

(iv)$\rightarrow$(i) Again prove the contrapositive. Assume there is a basis matrix B such that $B^{-1}b \geq 0$ but B^{-1} is not binary in column i even though $b_i > 0$. W.l.o.g. the nonbinary element is in column $i = m$. Then $B^{-1}e^m$ corresponds to a nonnegative solution where there is a hyperarc (J, k) with integral flow greater than one. Since the hypergraph is gainfree, by Theorem 2.6 the support is a hyperforest where each component is a hyperarboresence. With $b = e^m$ it follows that the support is actually a single hyperarboresence directed into vertex m. Then we may assume that the vertices in the support are ordered such that the head vertex always has a higher number than the tail vertices. Take k to be the highest numbered vertex such that (J, k) is not one. Note that $k < m$ because the unique hyperarc into m must have a flow of 1 to balance $b = e^m$. Outbound hyperarcs at k must have unit flows because they lead to higher numbered vertices. Thus, since tail weights are restricted to 1, it follows that there are at least two distinct hyperarcs directed out of vertex k in the support, and so two distinct paths of nondegenerate hyperarcs leading from vertex k to vertex m. Since the support is a hyperarboresence, the two paths must merge at vertex $l \leq m$, in a hyperarc (T, l) with one of distinct tails t_1, $t_2 \in T$ belonging to each path. Finally note that the support must contain a path from a source vertex s to vertex k in order for the flow on (J, k) to be positive. This path cannot use vertex l since the support is a hyperarboresence. Therefore $I_l[t_1] \cap I_l[t_2]$ contains s, and the hypergraph is not support disjointly reachable. □

Corollary 4.6. *If the Leontief directed hypergraph $G = (V, H)$ with constraint matrix A consisting of $0, \pm 1$ is either free of paracycles or disjointly reachable, then extreme flows are binary for every unit vector $b = e^i$ for which there is a feasible flow.*

Proof. A hypergraph with no paracycles can have none in a support. Also, if a Leontief directed hypergraph with A consisting of $0, \pm 1$ is disjointly reachable, it is support disjointly reachable. Thus the corollary follows by (iii) and (iv) of Theorem 4.5 respectively. □

Corollary 4.7. *If the Leontief directed hypergraph $G = (V, H)$ with Leontief constraint matrix A consisting of $0, \pm 1$ is either free of paracycles or disjointly reachable, then every basis submatrix feasible for some $b \geq 0$ has an inverse that can be permuted into a lower triangular, binary matrix.*

Proof. When A is Leontief and consists of $0, \pm 1$, Theorem 4.3 shows the inverse of any feasible basis is integral and permutable to lower triangular form. As with Corollary 4.6, if G is free of paracycles or disjointly reachable, either (iii) or (iv) of Theorem 4.5 also holds. Noting that A Leontief means every row is nontrivial, and so any b_i can be taken as positive in a basic feasible solution, part (i) of Theorem 4.5 than proves the inverse is all binary. □

Corollaries 4.6 and 4.7 imply being disjointly reachable is sufficient for a $0, \pm 1$ Leontief substitution flow problem over a Leontief matrix A to have binary feasible basis inverses and binary vertices for every unit vector right hand side. We summarize this and other results in Table 1.

Table 1
Summary of properties for Leontief flow problems on $G = (V, H)$ with incidence matrix A Leontief

Sufficient condition	Value iteration	Basis matrices	Basis inverses	Extreme solutions
Leontief substitution ($b \geq 0$)	asymptotically convergent in many cases (see Koehler et al., 1975)	Leontief	nonnegative	—
Leontief substitution ($b \geq 0$) and gainfree	strongly polynomial	Leontief and triangular	nonnegative and triangular	acyclic support
Leontief substitution ($b \geq 0$) and A integer (thus gainfree)	strongly polynomial	Leontief, triangular and integer	nonnegative, triangular and integer	acyclic support and integer for integer b
Leontief substitution ($b \geq 0$), A is $0, \pm 1$ (thus gainfree and integer), G disjointly reachable or free of paracycles	strongly polynomial	Leontief, triangular and $0, \pm 1$	triangular and binary	acyclic support and binary for unit vector b

Although disjoint reachability (as opposed to support disjoint reachability) is not necessary for binary results, this sufficient condition is significant because the property of being disjointly reachable can be tested in strongly polynomial time on gainfree hypergraphs. First, degenerate hyperarcs (J, i) are identified and deleted using the algorithms of Section 3. From the G that remains, construct an ordinary graph G' on the same vertex set, with one arc (j, i) for each $j \in J$ of each (J, i) in G. Standard reachability algorithms applied to variants of G' with each vertex k deleted in turn can now yield the sets $I_k[l]$ of (14). Definition (15) is then easily verified. We do not know whether there is a polynomial algorithms for testing support disjoint reachability.

Earlier work by Martin, Rardin and Campbell (1990) established that a sufficient condition for the binary property (ii) of Theorem 4.5 in acyclic Leontief directed hypergraphs is that there exist finite sets $\{R[i] \mid i \in V\}$ satisfying

$$R[j] \subseteq R[k] \quad \text{for all } (J, k) \in H,\ j \in J, \tag{16}$$

and

$$R[j_1] \cap R[j_2] = \emptyset \quad \text{for all } (J, k) \in H,\ j_1 \neq j_2 \in J. \tag{17}$$

Certainly (16) implies $R[s] \subseteq R[k]$ for each source vertex s with a path to k. Thus (17) requires that the index sets of source vertices with paths to distinct tails of hyperarcs (J, k) should not intersect. In the acyclic case this is exactly disjoint reachability in G.

5. Applications

Given their very special structure, it is surprising that gainfree, disjointly reachable, Leontief substitution flow problems arise in so many different settings. In the subsections below we briefly review some of those application areas, show how previous results relate to hypergraph results in Sections 2–4, establish some new results, and highlight open issues.

5.1. Generalized network flows

The most famous class of Leontief flow problems are the *network flow problem* and the *generalized network flow problem*, which have at most one negative entry per column of constraint matrix A as well as at most one $+1$. Ordinary network flows are the case with the negative entries -1; generalized flows allow arbitrary tail weights. Thus, in graph terms, a network flow is a Leontief flow problem on an ordinary directed graph, and a generalized flow problem is a Leontief flow on a directed generalized graph.

If a Leontief directed hypergraph has a paracycle, it must have a hyperarc with at least two tails. Since this cannot occur in either ordinary or generalized network flows, these Leontief flow problems are always disjointly reachable. Ordinary flow

problems are also gainfree because every directed cycle has a gain $=1$. Generalized network flows may or may not be gainfree.

The ordinary and generalized network flow cases of Leontief substitution flows are, of course, those with $b \geqslant 0$. To relate them to a more familiar category, recall the shortest path problem of finding a directed path from vertex s to vertex t in a given graph (or generalized graph) that is shortest in the sense of satisfying a unit demand at t. It is well known in the ordinary graph case that if there is no directed cycle of negative total cost, this problem can be resolved by solving a Leontief flow problem with a source arc inbound at s and a $b = e^t$. Extension to the gainfree generalized case leads to a converse characterization of all $b \geqslant 0$ cases on graphs or gainfree generalized graphs.

Lemma 5.1. *Every Leontief substitution flow problem on a directed graph or gainfree generalized graph that has a finite optimal solution can be solved by a series of at most m shortest path problems, where m is the number of vertices.*

Proof. Since every ordinary directed graph is gainfree, we need only treat gainfree generalized graphs. Consider a Leontief substitution flow problem on a gainfree generalized graph. Modify the problem by adding a new vertex s, replacing all source arcs $(\emptyset, k)$ by arcs (s, k), and creating a new source arc $(\emptyset, s)$. One way to solve this Leontief flow problem is to solve separate subproblems for each $b_i > 0$. Each such problem has a unit demand at vertex t and $b_i = 0$ for $i \neq t$. If x^t is the resulting optimal flow, $x = \sum b_t x^t$ solves the full problem. But because the associated generalized graphs are gainfree, and the only source vertex is s, the support of each x^t will consist of a shortest path from s to t. Thus, this scheme for the given Leontief substitution problem is equivalent to solving up to m shortest path problems and summing results. □

Cosares and Adler (1987) have recently shown that ordinary network flow problems with $b \geqslant 0$ also have a number of elegant properties in the context of solution by the dual Simplex algorithm. Among other things, the dual simplex is polynomial using Dantzig's (least $\bar{b}$) pivoting rule, and the famous Hirsch conjecture about the underlying dual polytope is proved to hold. Cosares and Adler also suggest that their results may extend to some generalized network flows. Since the Hirsch conjecture is known to hold for arbitrary Leontief substitution flows (Grinold, 1971), these results raise the possibility that some form of simplex algorithm is polynomial on at least gainfree Leontief substitution flow problems.

5.2. Expert systems with Horn clause knowledge bases

A *Horn clause* in predicate logic is a clause with positive predicates and at most one conclusion. Horn clauses are an integral part of logic programming languages, such as Prolog, which are used in many artificial intelligence applications. We show how to model as a gainfree Leontief substitution flow, the problem of deciding the

truth of a unary predicate bound to a constant in a Horn clause knowledge base with only unary predicates. The procedure is motivated with the following example taken from Walker, McCord, Sowa and Wilson (1987).

Washable allergenic things are washed. Nonwashable allergenic things are vacuumed. Everything that is gray and fuzzy is allergenic. Shirts and dogs are washable. Lamps, sofas and cats are nonwashable. Do I vacuum my gray and fuzzy cat Tiger?

In logic programming terms, this example is described by the following:

Predicates

```
washable(X).    allergenic(X).  thing(X).  washed(X).
nonwashable(X). vacuumed(X).    gray(X).   fuzzy(X).
shirt(X).       dog(X).         lamp(X).   sofa(X).
cat(X).
```

Clauses:

```
washed(X)      :- washable(X),     allergenic(X),  thing(X).
vacuumed(X)    :- nonwashable(X),  allergenic(X),  thing(X).
allergenic(X)  :- gray(X),         fuzzy(X),       thing(X).
washable(X)    :- short(X).
washable(X)    :- dog(X).
nonwashable(X) :- lamp(X).
nonwashable(X) :- sofa(X).
nonwashable(X) :- cat(X).
```

Facts:

```
cat(tiger).  gray(tiger).  fuzzy(tiger).  thing(tiger).
```

Constants:

```
tiger.
```

Goal:

```
vacuumed(tiger).
```

The following method models this same example as a gainfree Leontief substitution flow problem.

- Define a vertex in the hypergraph for each predicate.

- Define a hyperarc for each clause. Do so by directing the head of the hyperarc into the vertex which corresponds to the conclusion of the clause. The tails of the hyperarc are connected to the conditions of the clause. Hyperarcs with multiple tails and heads are required in the non-Horn case.
- Bind the facts to the appropriate predicates. Do so by defining a vertex for each fact and create an arc (predicate(constant), predicate(X)) for each fact-predicate pair. This may seem redundant, however it allows for testing of other predicates bound to a constant with little modification of the hypergraph. For example, to test washed(tiger) requires changing only the b vector and no changes to the A matrix.
- Create a source vertex v_{constant}, and hyperarc $(\emptyset, v_{\text{constant}})$ for the constant that is bound to the goal predicate. Then add the arcs from the constant vertex to the corresponding fact vertices. In this example there is a source vertex v_{tiger} and arcs from vertex v_{tiger} to the fact vertices thing(tiger), cat(tiger), gray(tiger) and fuzzy(tiger).
- The only demand vertex is the goal predicate vertex v_{goal} with a demand of one unit. If the problem is feasible then the goal predicate is true; if the problem is infeasible the goal predicate is false.

Figure 4 illustrates this process for the example problem. The issue of determining which predicates are an implication of the facts in the knowledge base reduces to the distinction between Leontief and pre-Leontief matrices made earlier. A predicate is provable exactly when its vertex is nontrivial, using the construction given above.

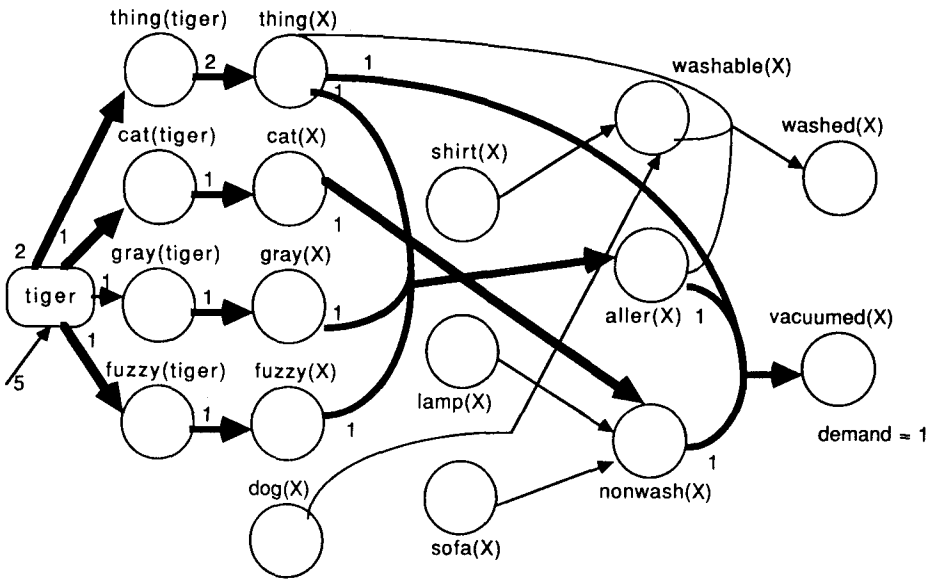


Fig. 4. Hypergraph for knowledge base.

The Leontief flow problem is gainfree because all the hyperarc multipliers are +1. Since the right hand side is the unit vector, by Theorem 4.2 the Leontief flow problem for determining which predicates follow from the facts has integral solutions. See Jeroslow and Wang (1989) for an earlier proof of this result. However, the solutions are not necessarily binary. A positive integer greater than one in a solution corresponds to that hyperarc being used more than once in the proof of a proposition. In Figure 4 the numbers above the hyperarc correspond to the number of times the hyperarc is used in the proof of vacuum(tiger), i.e. its optimal solution value. The highlighted hyperarcs indicate the solution.

If the methods of Section 3 were applied directly to logic tasks like the above example, Value Iteration would first perform a forward pass to settle whether the goal is provable. (The forward pass algorithm is akin to forward chaining, see for example, Dowling and Gallier (1984).) Then, if so, Primal Retrieval is invoked to exhibit a proof. This approach has the virtue of precision. Still, for the large knowledge bases of expert systems, with their possible exponential numbers of bindings (complete assignment of constants to variables), such a scheme is often impractical. In the example above each predicate has only one variable. Thus, there is a unique assignment of constants (tiger) to variables and a simpler reduction to propositional logic. In cases where there is more than one variable per predicate (such as in recursion) the assignment of constants becomes a problem. See Chandru and Hooker (1988) and Jeroslow (1985) for a discussion of mathematical programming approaches to first order predicate logic.

Logic programming languages like Prolog deal heuristically with this difficulty by *backward chaining*, i.e. searching backwards from the goal to look for a proof. The idea is analogous to applying Primal Retrieval without first having used Value Iteration to fix the labels $J[i]$. Infinite loops can certainly occur. (For example, in some implementations of backward chaining a hypergraph with cycles may result in an infinite loop, depending upon the order in which the rules are entered.) In fact, the question of whether the search will terminate is undecidable for general cases. Still, sufficient conditions based on structural properties of what we have viewed as the underlying hypergraph can be obtained. The recent work of Ulman and Van Gelder (1988) describes some approaches.

5.3. Polyhedral characterizations for recursively defined graphs

Much of the recent interest in Leontief-like models centers on their ability to characterize the solution sets of combinatorial problems in terms of the extreme solutions of a suitable Leontief substitution flow problem. One family of cases arises in the context of operations management. (See Martin, Rardin and Campbell (1990).)

A much broader class of examples arises in the context of optimization over recursively defined families of graphs. Recent papers by Wimer, Hedetniemi and Laskar (1985) and Bern, Lawler and Wong (1985) summarize a large list of recursively defined graph forms, and a larger list of optimization problems, each combination of which is polynomially solvable by a variety of discrete dynamic programming.

Graphs $G(V, E)$ belonging to one of these classes are viewed as having a fixed number $\kappa \stackrel{\text{def}}{=} |K|$ distinguished vertices $K \subset V$ called *terminals*. Each family $\mathcal{F}$ begins from a set $\mathcal{F}_0 \subset \mathcal{F}$ of *primitive* starting graphs. Other members are obtained by repeated application of a finite list of *composition* operations defined in terms of the terminals. Compositions combine two members $G^i = (V^i, E^i)$, $G^j = (V^j, E^j)$ of $\mathcal{F}$ into a new member, G^k of $\mathcal{F}$ by $V^k \leftarrow V^i \cup V^j$, $E^k \leftarrow E^i \cup E^j$, and then constructing terminal set K^k by

(i) identifying some of the terminal vertices (by identify, we mean treat as the same vertex) in $K^i \subseteq V^i$ with those of $K^j \subseteq V^j$,

(ii) adding new edges or deleting edges between terminals in K^i and ones in K^j,

(iii) adding new edges or deleting edges between two terminals in K^i, or two terminals in K^j, and

(iv) selecting κ terminals K^k from $K^i \cup K^j$.

One famous class of recursively defined graphs are *series-parallel* graphs with $\kappa = 2$ terminals. Primitives for this family of graphs are single edges (with their two ends defining terminals). *Series* composition combines two series-parallel graphs by identifying a terminal of one with a terminal of the other; the nonidentified terminals define the terminal vertices of the result. *Parallel* composition identifies both terminals of the two graphs, the combined terminals being the terminals of the result. See Figure 5.

Optimization algorithms on κ-terminal graphs such as series-parallel graphs are organized around *states* of partial solution of subgraphs encountered in composing

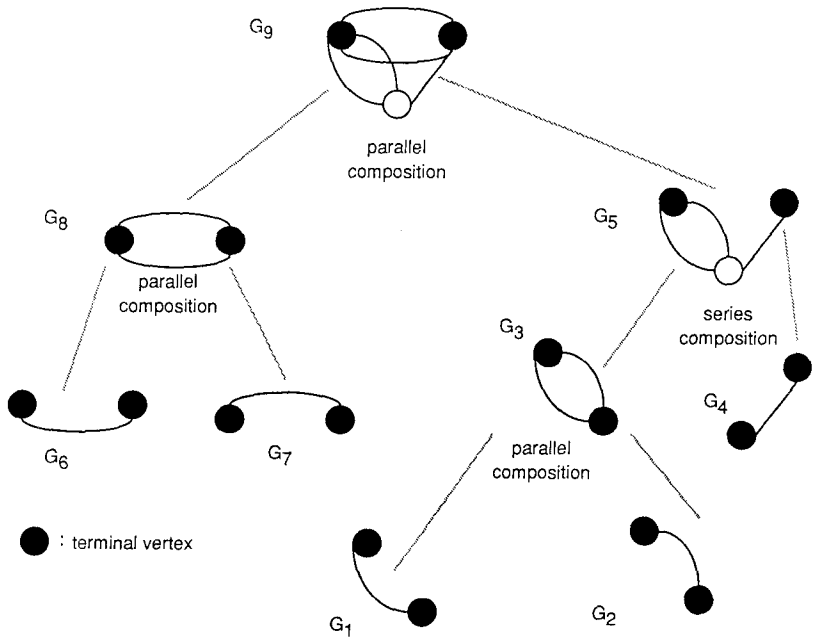


Fig. 5. Recursive composition of a series-parallel graph.

the one of interest. In some cases states reflect portions of an optimal solution contained in the subgraph; in others they account for forbidden substructures. Algorithms proceed in "bottom up" fashion, first solving all states of primitives, then recursively finding the best way to reach all states of each intermediate subgraph.

An easy series-parallel example is the problem of finding the minimum weight cycle. For each subgraph in the composition there are two states. A subgraph is in the *cycle state* if it contains the minimum cost cycle. A subgraph is in the *path state* if it contains a partial cycle in the form of a path connecting the terminal vertices. A subgraph in the cycle state and a subgraph in the path state cannot be combined in either a parallel or series composition. A composition of two subgraphs, which are both in the cycle state, is not allowed. Two subgraphs in the path state joined by a series operation yield a subgraph in the path state. Two subgraphs in the path state joined by a parallel operation yield a subgraph in the cycle state.

Figure 6 depicts the optimization through cycle and path states for the graph composed in Figure 5. For example, in the parallel composition of G^9, the cycle state can be reached by inheriting a cycle from either of the subgraphs G^5 and G^8, or by combining the path states of the two subgraphs. In the series composition of G^5, path states of G^3 and G^4 combine to produce a path in the composition.

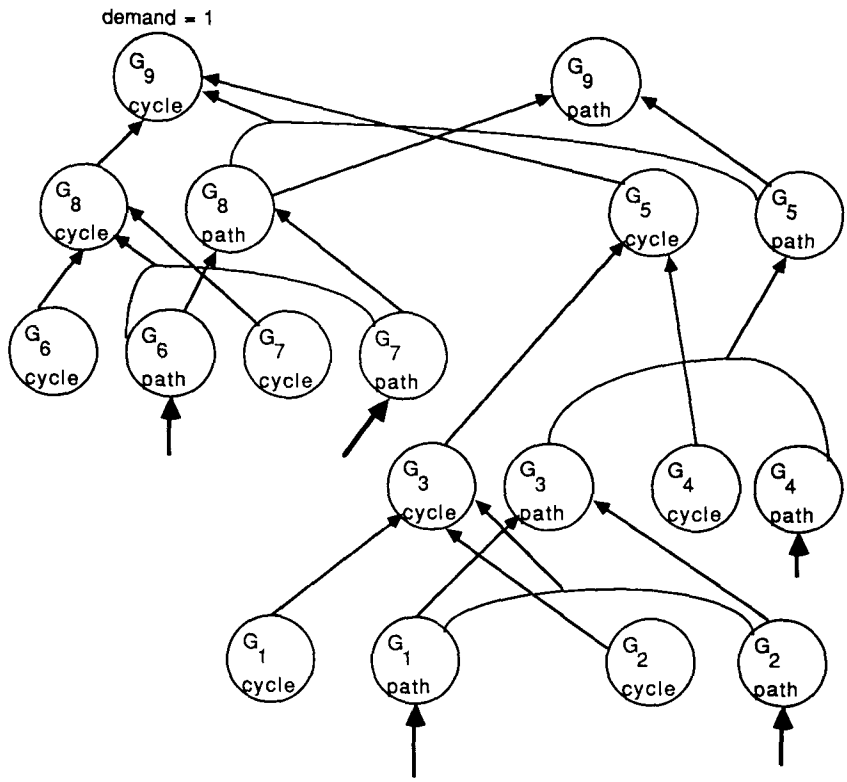


Fig. 6. State composition hypergraph for minimum cycle on the graph of Figure 5.

If $Z \subseteq \mathbb{R}^n$ and CONV(Z) is a polyhedron, then:

$$Ax + Dz = b, \quad x \geq 0,$$

is an *extended polyhedral representation* of CONV(Z) if

$$\text{CONV}(Z) = \{z \mid \text{there exists } x \geq 0 \text{ such that } Ax + Dz = b\}.$$

Such formulations are valuable because use of the *auxiliary* variables x often makes it possible to obtain a *compact* (polynomial in n) characterization of CONV(Z) sufficiently small to be inserted in sharpening the linear programming relaxation of a model with more complex constraints (see Martin, 1987; and Eppen and Martin, 1987).

Martin, Rardin and Campbell (1990) show that Leontief flow problems over state composition graphs like the one in Figure 6 lead directly to extended polyhedral characterizations of optimization problems solvable on recursively defined graphs. The demand at every vertex is zero except for a +1 at the goal state of the final graph (the *cycle* state of G^9 in Figure 6). Vertices that correspond to a primitive edge in a path state are given source hyperarcs (bold hyperarcs in Figure 6) directed into the vertex; with a the flow of one unit on the source hyperarc indicating that the corresponding edge is in the cycle (these source hyperarcs correspond to the original z variables). The remaining hyperarcs constitute the auxiliary x variables.

To provide a correct extended polyhedral representation, the extreme flows of such Leontief formulations must be binary in x and z. Fortunately, the models satisfy virtually all of the desirable properties highlighted in Sections 2 and 4. The demand vector is a unit vector, hence it is a Leontief substitution flow problem. It is also gainfree since all coefficients are 0 or ± 1. The recursive nature of the algorithms they model also assures the Leontief directed hypergraphs are acyclic (although this is not necessary or sufficient for binary integrality).

The more subtle issue is to establish that the hypergraphs are disjointly reachable (i.e. they contain no paracycles), so that Corollary 4.6 applies. To see that this is true, we focus on the edge sets in composition rules (i)-(iv) above. The fact that edge sets of composed subgraphs are always disjoint means the subgraphs could not have a common ancestor vertex in the Leontief hypergraph. Paracycles are thus impossible.

5.4. Polyhedral representation of send and split

Erickson, Monma and Veinott (1987) present a very elegant algorithm, the *send-and-split method*, for optimizing minimum concave cost network flows. By using auxiliary variables, we show how to model the send and split algorithm as a gainfree Leontief substitution flow problem. Then, using the binary integrality results of Section 4 we show that the Leontief directed hypergraph formulation is a correct polyhedral representation of the associated concave network flow problem and allows it to be solved as a linear program.

For sake of simplicity, we assume that the network flow problem under consideration is a single source, uncapacitated problem. The method of Wagner (1959) can be used to reduce problems with capacities and multiple sources into this format. The problem under consideration on digraph (N, E) with vertex set N and arcs E is model (NF):

$$\text{(NF)} \quad \min \sum_{(i,j)\in E} c_{ij}[z_{ij}] \tag{18}$$

$$\text{s.t.} \quad \sum_{(j,l)\in E} z_{jl} - \sum_{(l,j)\in E} z_{lj} = \begin{cases} -\sum_{i\in D} d_i, & \text{for } l = i_0, \\ d_l, & \text{all } l \in D, \\ 0, & \text{all } l \in T, \end{cases} \tag{19}$$

$$z_{ij} \geqslant 0, \quad \text{all } (i,j) \in E. \tag{20}$$

In this formulation, i_0 indexes the supply vertex, T indexes pure transshipment vertices and D indexes the demand vertices where d_j is the nonnegative demand at vertex j, z_{ij} is the flow on arc (i,j), and $c_{ij}[z_{ij}]$ is the cost of sending z_{ij} units of flow over arc (i,j). The $c_{ij}[*]$ are assumed to be concave on the nonnegative real line with $c_{ij}[0] = 0$. We assume that there is a minimum cost flow for some demand vector $(d_j)_{j\in D}$. Then from Erickson et al. (Theorem 1) there is a minimum cost flow for which the induced subgraph is a forest. Therefore, in an extreme flow $\bar{z}$, for every arc (i,j) there exists an $I \subseteq D$ such that

$$\bar{z}_{ij} = 0 \quad \text{or} \quad \bar{z}_{ij} = \sum_{l\in I} d_l. \tag{21}$$

In words, the flow on arc (i,j), in an extreme flow, is equal to zero, or the sum of the demands for some subset of demand vertices.

The property given in (21) simplifies solution considerably since a flow choice for (NF) consists of two types of decisions. Given that we are at vertex i with a flow of $\sum_{l\in I} d_l$, one possibility is to *send* the entire flow on to an adjacent vertex j at cost $c_{ij}[\sum_{l\in I} d_l]$. The other option is to *split* the flow into proper subsets $\sum_{l\in I'} d_l$ and $\sum_{l\in I\setminus I'} d_l$, both now located at the same vertex i.

The send-and-split algorithm resolves these decisions in backwards fashion starting from singleton demand sets $\{k\}$. Shortest path computation with costs $c_{ij}[d_k]$ establishes the cost of sending this flow from each possible split point to destination k. We now know the least cost way of completing some demand sets given that they are at any particular vertex. Proceeding recursively, we can determine similar costs of completing a larger demand set I by picking a minimum cost pattern of the form: send the combined flow to a vertex over a shortest path with costs $c_{ij}[\sum_{l\in I} d_l]$, split into I' and $I\setminus I'$, and finish in the optimal manner. The result is a sequence of shortest path problems with different costs.

The Leontief flow model of this computation parallels the backwards solution. Specifically, it is formed as follows:

- Create vertex set $W = N \times P(D)$ for the hypergraph where $P(D)$ is the power set of the demand vertices.

- Create a *split hyperarc* for each possible split decision. These hyperarcs have head vertex (i, I) and tail vertices $\{(i, I'), (i, I \backslash I')\}$. Each split hyperarc always has two tails and the tail weights are always +1. A cost of zero is given to each split hyperarc in the objective function.
- Create a *send hyperarc* for each possible send decision. For each $(i, j) \in A$ and $I \subseteq D$ the send hyperarc has head (i, I) and tail (j, I). Note the hyperarc points in the reverse direction of the network arc (i, j). Tail weights are +1. The cost assigned the send hyperarc in the objective function is $c_{ij}[\sum_{h \in I} d_h]$.
- Create a source hyperarc directed into every vertex $(i, \{i\})$ all $i \in D$. These hyperarcs are also assigned a zero cost in the objective function.
- Create a unit demand at vertex (i_0, D).

This process is illustrated for the network flow problem in Figure 7. Figure 8 contains the corresponding Leontief directed hypergraph. Notice that directions of arcs in Figure 8 are counter to the direction of arcs in Figure 7.

The values of the flow variables are recovered from the values of the hyperarc variables as follows:

$$\sum_{I \subseteq D} \left[\sum_{h \in I} d_h \right] x[(j, I), (i, I)] \to z_{ij}.$$

Combining with the Leontief flow balance constraints gives the reformulated or

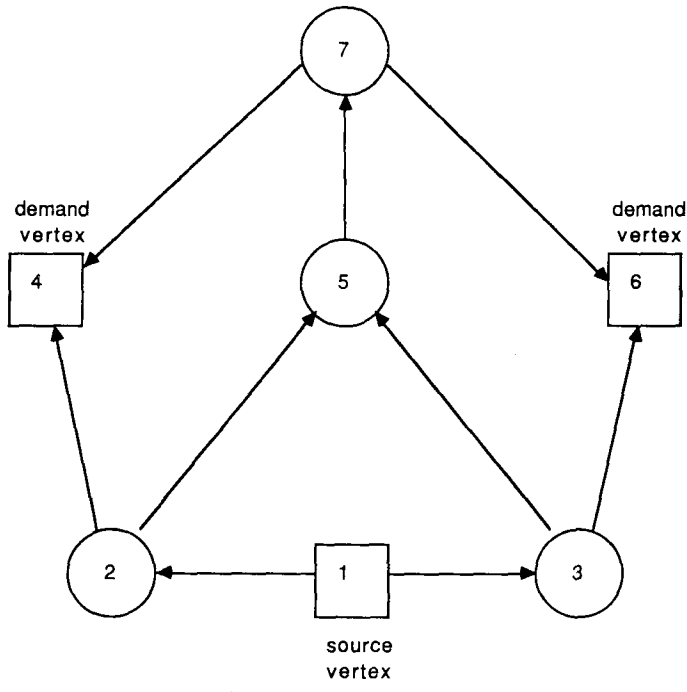


Fig. 7. Concave network flow example.

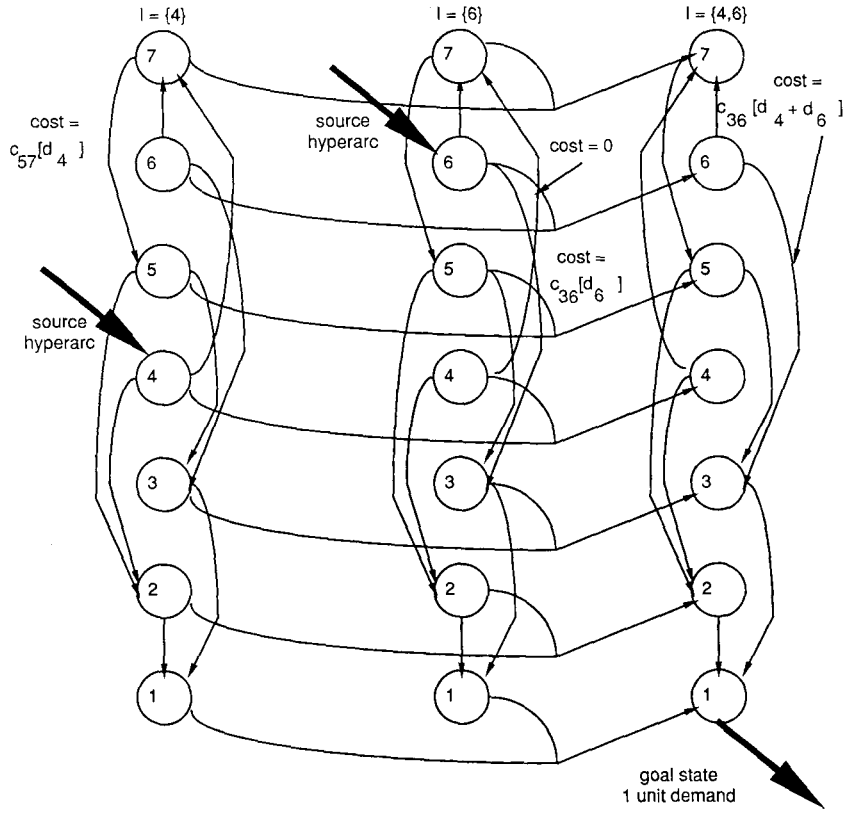


Fig. 8. Send-and-split representation of network flow problem in Figure 7.

extended polyhedral representation (RNF):

$$\text{(RNF)} \quad \min \quad \sum_{I \subseteq D} \sum_{(i,j) \in E} c_{ij} \left[\sum_{h \in I} d_h \right] x[(j, I), (i, I)], \tag{22}$$

$$\text{s.t.} \quad \sum_{(i,j) \in E} x[(j, I), (i, I)] + \sum_{I' \subset I} x[\{(i, I'), (i, I \backslash I')\}, (i, I)]$$

$$- \sum_{(j,i) \in E} x[(i, I), (j, I)] - \sum_{I \subset H} x[\{(i, I), (i, H \backslash I)\}, (i, H)]$$

$$= \begin{cases} 1, & (i, I) = (i_0, D), \\ -x[\emptyset, (i, I)], & (i, I) = (i, \{i\}), \\ 0, & \text{all other } (i, I), \end{cases} \tag{23}$$

$$\text{all } x[(j, I), (i, I)], x[\{(i, I'), (i, I/I')\}, (i, I)] \geqslant 0, \tag{24}$$

$$\sum_{I \subseteq D} \left(\sum_{h \in I} d_h \right) x[(j, I), (i, I)] = z_{ij} \quad \text{for all } (i, j) \in E. \tag{25}$$

Lemma 5.2. *The hypergraph of the Leontief flow problem defined by* (22)-(24) *is disjointly reachable.*

Proof. The only hyperarcs with multiple tails are $(\{(i, I'), (i, I\backslash I')\}, (i, I))$. It suffices to show that if $h \in I'$, then there is no path from the vertex $(h, \{h\})$ to vertex $(i, I\backslash I')$. This is trivial because, by construction of the hypergraph, there is a path from vertex $(h, \{h\})$ to vertex (i, I') if and only if $h \in I'$. □

The binary integrality of (RNF) and a result similar to Theorem 5.3 was first proved in Campbell (1987). However, that proof is specific to the structure of (RNF). We are now in a position to establish it much more easily.

Theorem 5.3. *The formulation given by* (23)-(25) *is an extended polyhedral representation of the formulation given by* (19)-(20).

Proof. Show $\bar{z}$ is a feasible solution to (NF) if and only if there exists $\bar{x}$ such that $(\bar{z}, \bar{x})$ is a feasible solution to (RNF). First show $(\bar{z}, \bar{x})$ feasible in (RNF) implies $\bar{z}$ is feasible to (19)-(20). To see this, create an aggregate constraint for each $l \in N$, by summing the all constraints in (23) for $i = l$ using multipliers of $-(\sum_{h \in I} d_h)$. With this weighting all split variables cancel in the aggregate constraint leaving

$$\sum_{(j,l)\in E} \sum_{I \subseteq D} \left(\sum_{h \in I} d_h \right) \bar{x}[(l, I), (j, I)] - \sum_{(l,j)\in E} \sum_{I \subseteq D} \left(\sum_{h \in I} d_h \right) \bar{x}[(j, I), (l, I)]$$
$$= \begin{cases} -\sum_{h \in D} d_h, & l = i_0, \\ d_l \bar{x}[\emptyset, (l, \{l\})], & l \in D, \\ 0, & l \in T. \end{cases} \tag{26}$$

To see that the right hand sides of (26) are correct observe that the right hand sides of (23) are all 0 except when $l = i_0$, or $l \in D$. The first of these produces $-\sum_{h \in D} d_h$ because of the unit demand at vertex (i_0, D). For $l \in D$, the right hand side in the aggregate constraint is $d_l \bar{x}[\emptyset, (l, \{l\})]$. By Lemma 5.2, the fact that all nonzero coefficients in (23)-(24) are ± 1 and Corollary 4.6 ecery extreme point of the polyhedron defined by (23)-(24) is binary with support defining a hyperarboresence. In every support hyperarboresence (extreme point) every source hyperarc $(\emptyset, (l, \{l\}))$ has a flow of one unit. In the hypergraph corresponding to (23)-(24) every directed cycle contains only send hyperarcs (which have a single tail), hence no source hyperarcs are in any directed cycle and $x[\emptyset, (l, \{l\})] = 0$ in every extreme direction of recession. By the theorem of Minkowski every feasible $(\bar{z}, \bar{x})$ to (23)-(24) is a convex combination of extreme points and nonnegative combination of extreme directions of recession for the pointed polyhedron defined by (23)-(24). The fact that the source hyperarcs are zero in the extreme directions of recession and the fact that the flow on every source hyperarc is one in an extreme point solution implies that $\bar{x}[\emptyset, (l, \{l\})] = 1$ in every feasible solution. Therefore, $d_l \bar{x}[\emptyset, (l, \{l\})] = d_l$.

Then, using the definition of z_{ij} in (25) and recalling that send hyperflows x are reverse to z flows, we see that (26) implies (19).

Assume $\bar{z}$ is feasible in (19)-(20) and show that there exists $(\bar{z}, \bar{x})$ feasible in (23)-(25). Again use the theorem of Minkowski and show that for every extreme point and extreme direction of recession for the polyhedron defined by (19)-(20) there is a corresponding point in the polyhedron defined by (23)-(25). It is clear that for every combination of send and split decisions producing an extreme point solution in (NF) there is a corresponding support hyperarboresence of an extreme point of the polyhedron defined by (23)-(25). Also, every extreme direction of recession of the polyhedron defined by (19)-(20) corresponds to a circulation with support which is a directed cycle. For all $I \subseteq D$, each of these directed cycles also exist in the hypergraph (in reverse direction). Hence, there is a set of send decisions corresponding to any circulation in (NF). □

Although both (NF) and (RNF) have feasible regions which are polyhedrons model (NF) is a *concave* network optimization problem. Model (RNF) is a *linear* program. In addition to the concavity of $c_{ij}[*]$, it is crucial that the x variables in (23)-(25) be binary in order for the objective function (22) to correctly model (18).

Also note that formulation (RNF) is not polynomial in the number of demand vertices because all subsets of D have to be considered. However, Erickson et al. (1987) show that when the graph of (NF) can be embedded in the plane with vertices of $i_0 \cup D$ on at most a constant number of β faces, many subsets can be disregarded. In particular, attention can be restricted to $O(\beta|D|^2)$ demand subsets, and (RNF) becomes a polynomial size extended formulation.

5.5. *Leontief flow model of deterministic Turing machines and implications*

Jones and Laaser (1974) give a model of a deterministic Turing machine in conjunctive normal form, CNF, where each disjunctive clause of literals contains at most one positive literal (i.e., is Horn). (See also Dobkin, Lipton and Reiss (1979), who use this result to show linear programming is log-space hard for $\mathcal{P}$ — and now, given Khachian (1979) is log-space complete.) By slightly modifying the Jones and Laaser result, and using the material in Section 5.2 on expert systems, we show how to model a deterministic Turing machine as a gainfree Leontief substitution flow problem.

Consider a single tape Turing machine with a finite alphabet $\Lambda = \{0, 1, \not b\}$, finite states Q and next move function $\delta : Q \times \Lambda \to Q \times \Lambda \times \{-1, 0, 1\}$. To model a Turing machine which accepts language L as a satisfiability problem on a Horn clause knowledge base, define the atomic propositions:

- $v(t, c, q, \alpha)$ is true if, before move t, and after move $t-1$, the machine is in state $q \in Q$ with symbol $\alpha \in \Lambda$ in tape square c and the tape head is over square c.
- $w(t, c, \alpha)$ is true if, before move t, and after move $t-1$, tape square c contains symbol α and the tape head is not over square c.

The truth of an atomic proposition involving tape square c at move t is completely determined by the next move function, the state and the contents of tape squares $c-1$, c, and $c+1$ at move $t-1$. See Table 2 for a list of the necessary Horn clauses. We assume, without loss, that the input language L tested is a binary string, the length of the input string z, is n, the input string begins in square 1, the time complexity is $\phi(n)$, $q_0 \in Q$ is the state of the machine at time zero, $q_A \in Q$ is the accepting state where $z \in L$, the total number of tape squares required (both input and workspace) is $\nu(n)$, and the tape head is initially over square 1.

Atomic propositions which do not use tape squares 1 through $\nu(n)$ are not included in the knowledge base.

Theorem 5.4. *Let T be a deterministic Turing machine for deciding if a string z is in language L. Then z is accepted by the Turing machine if and only if an atomic proposition*

Table 2
Horn clause knowledge base for deterministic Turing machines

Rules	
$w(t, c, \alpha)$:-	$w(t-1, c-1, \alpha_{c-1})$, $w(t-1, c, \alpha)$, $w(t-1, c+1, \alpha_{c+1})$ for all $t, c, \alpha, \alpha_{c-1}, \alpha_{c+1}$
$w(t, c, \alpha)$:-	$v(t-1, c-1, q', \alpha_{c-1})$, $w(t-1, c, \alpha)$, $w(t-1, c+1, \alpha_{c+1})$ for all $t, c, \alpha, \alpha_{c+1}$, $(q', \alpha_{c-1}) \in \delta^{-1}(Q \times \Lambda \times \{-1, 0\})$
$w(t, c, \alpha)$:-	$w(t-1, c-1, \alpha_{c-1})$, $v(t-1, c, q', \beta)$, $w(t-1, c+1, \alpha_{c+1})$ for all $t, c, \alpha, \alpha_{c-1}, \alpha_{c+1}$, $(q', \beta) \in \delta^{-1}(Q \times \{\alpha\} \times \{-1, 1\})$
$w(t, c, \alpha)$:-	$w(t-1, c-1, \alpha_{c-1})$, $w(t-1, c, \alpha)$, $v(t-1, c+1, q', \alpha_{c+1})$ for all $t, c, \alpha, \alpha_{c-1}$, $(q', \alpha_{c+1}) \in \delta^{-1}(Q \times \Lambda \times \{0, 1\})$
$v(t, c, q, \alpha)$:-	$v(t-1, c-1, q', \alpha_{c-1})$, $w(t-1, c, \alpha)$, $w(t-1, c+1, \alpha_{c+1})$ for all $t, c, q, \alpha, \alpha_{c+1}$, $(q', \alpha_{c-1}) \in \delta^{-1}(\{q\} \times \Lambda \times \{1\})$
$v(t, c, q, \alpha)$:-	$w(t-1, c-1, \alpha_{c-1})$, $v(t-1, c, q', \beta)$, $w(t-1, c+1, \alpha_{c+1})$ for all $t, c, q, \alpha, \alpha_{c-1}, \alpha_{c+1}$, $(q', \beta) \in \delta^{-1}(\{q\} = \{\alpha\} \times \{0\})$
$v(t, c, q, \alpha)$:-	$w(t-1, c-1, \alpha_{c-1})$, $w(t-1, c, \alpha)$, $v(t-1, c+1, q', \alpha_{c+1})$ for all $t, c, q, \alpha, \alpha_{c-1}$, $(q', \alpha_{c+1}) \in \delta^{-1}(\{q\} \times \Lambda \times \{-1\})$
Facts	
$v(1, 1, q_0, 1) \leftrightarrow z_1 = 1$ $v(1, 1, q_0, 0) \leftrightarrow z_1 = 0$	
$w(1, c, 1) \leftrightarrow z_c = 1$, $c = 2, \ldots, n$ $w(1, c, 0) \leftrightarrow z_c = 0$, $c = 2, \ldots, n$	
$w(1, c, \not b)$ $c = n+1, \ldots, \nu(n)$	

corresponding to the accepting state can be set true in the Horn clause knowledge base given by Table 2.

Proof. It suffices to show for the Horn clause knowledge base of Table 2 that (i) for each move t exactly one $v(t, c, q, \alpha)$ is true, and (ii) that for each tape square c and each move t at most one of the atomic propositions $w(t, c, \alpha)$, $v(t, c, q, \alpha)$ is true. These two conditions require that the tape head be above a unique tape square at every move, the machine be in a unique state at every move, and each tape square contains a unique symbol at each move.

Proof by induction. For $t = 1$ the result is obvious since either $v(1, 1, q_0, 1)$ is true and $v(1, 1, q_0, 0)$ false when $z_1 = 1$, or $v(1, 1, q_0, 1)$ is false and $v(1, 1, q_0, 0)$ true when $z_1 = 0$. Similarly for the w atomic propositions.

Assume (i) and (ii) are true for moves 1 through t. First show part (i) for move $t+1$, i.e. distinct $v(t+1, c, q, \alpha)$ and $v(t+1, c', q', \alpha')$ cannot both be true. There are two subcases to consider: (i)(a) $c \neq c'$ or $q \neq q'$, and (i)(b) $c = c'$, $q = q'$, and $\alpha \neq \alpha'$. From Table 2 it follows that if $v(t+1, c, q, \alpha)$ is true there must be a corresponding v atomic proposition true at move t.

Similarly for $v(t+1, c', q', \alpha')$. But $c \neq c'$ or $q \neq q'$ and the fact that a deterministic Turing machine only allows unique next move functions implies that at least two distinct v atomic propositions must be true at move t which contradicts the induction hypothesis so (i)(a) cannot happen.

Next consider (i)(b) with $c = c'$, $q = q'$, but $\alpha \neq \alpha'$. If the tape head is to the left or right of c before move t and only one v atomic proposition is true, then from Table 2 both $w(t, c, \alpha)$ and $w(t, c, \alpha')$ are true which contradicts (ii) of the induction hypothesis. If the tape head is above square c at the start of move t then due to the uniqueness of the next move function, at least one of the w atomic propositions for square c must also be true in order to account for both α and α' occurring in true atomic propositions, again contradicting part (ii) of the induction hypothesis. The proof of part (ii) for move $t+1$ is similar. □

The hypergraph flow problem corresponding to the Horn clause knowledge base of Table 2 is

$$z_1 x[\emptyset, v(1, 1, q_0, 1)] - \sum \text{flow out variables} = 0, \tag{27}$$

$$(1 - z_1) x[\emptyset, v(1, 1, q_0, 0)] - \sum \text{flow out variables} = 0, \tag{28}$$

$$z_c x[\emptyset, w(1, c, 1)] - \sum \text{flow out variables} = 0, \quad c = 2, \ldots, n, \tag{29}$$

$$(1 - z_c) x[\emptyset, w(1, c, 0)] - \sum \text{flow out variables} = 0, \quad c = 2, \ldots, n, \tag{30}$$

$$x[\emptyset, w(1, c, b)] - \sum \text{flow out variables} = 0, \quad c = n+1, \ldots, \nu(n), \tag{31}$$

$$\text{hypergraph flow constraints for } t \geq 1, \tag{32}$$

$$\sum_{J} \sum_{c=1}^{\nu(n)} \sum_{\alpha \in A} x[J, v(\phi(n), c, q_A, \alpha)] = 1, \tag{33}$$

$$\text{nonnegativity.} \tag{34}$$

Constraint (34) is merely a unit demand requirement at the goal of accepting z. Because we do not know a priori whether $z_c = 0$ or $z_c = 1$, both source hyperarcs are provided in (27)-(30). Constraints (27)-(31) are the conservation of flow constraints for all atomic propositions at time $t = 1$.

Theorem 5.5. *For every language $L \in \mathcal{P}$ and every input $\bar{z} \in \{0, 1\}^n$ there exists a gainfree Leontief substitution flow problem with size polynomial in n which is feasible if and only if $\bar{z} \in L$.*

Proof. If $L \in \mathcal{P}$, then for any input z of size n there is a deterministic Turing machine T, with time and space complexity bounded by a polynomial to decide the membership of z in L. For any fixed n and Turing machine T, construct the corresponding Leontief flow constraints for the clauses of Table 2. If $\bar{z}_1 = 1$ then by (27) source hyperarc $(\emptyset, v(1, 1, q_0, 1))$ has no restriction of flow and $v(1, 1, q_0, 1)$ is a fact, while hyperarcs with tail vertices $v(1, 1, q_0, 0))$ must have zero flow in order to satisfy (28) so $v(1, 1, q_0, 0)$ cannot be used as a fact. Conversely, when $\bar{z}_1 = 0$ source hyperarc $(\emptyset, v(1, 1, q_0, 0))$ has no restriction of flow and $v(1, 1, q_0, 0)$ is a fact, while hyperarcs with tail vertices $v(1, 1, q_0, 1)$ must have zero flow and $v(1, 1, q_0, 1)$ cannot be used as a fact. Similarly for the hyperarcs with $c = 2, \ldots, n$. Clearly this Leontief flow problem is gainfree since all nonzeros are ± 1. It is a Leontief substitution flow with integral right hand side since the right hand side is the unit vector. Since $L \in \mathcal{P}$ both $\phi(n)$ and $\nu(n)$ are polynomial functions of n. The number of atomic propositions in the Horn clause system is then bounded by the polynomial $\phi(n) \times \nu(n) \times |Q| \times |\Lambda|$. Since every hyperarc has at most three tails the resulting formulation (27)-(34) is polynomial in size. By Theorem 5.4 it has a feasible solution, and hence integral solution, if and only if $\bar{z} \in L$. □

Theorem 5.5 says, that in a certain sense, gainfree Leontief substitution flow problems are sufficient to provide an extended characterization of every language $L \in P$. In contrast to Sections 5.3-5.4, however, we have not described a system of linear equations in z and x such that (z, x) is feasible if and only if $z \in \mathrm{CONV}(L)$. It is not necessarily true that if $(\bar{z}, \bar{x})$ is feasible to (27)-(34) then $\bar{z} \in \mathrm{CONV}(L)$. In Theorem 5.5 we construct a specific Leontief flow problem for each input z. To have a true extended representation, the z_i must be treated as variables. But then representation (27)-(34) is bilinear (linear only if z or x is fixed). Thus, we have proved every language $L \in P$ has an *extended bilinear representation*, but whether each has a linear one remains an open question.

6. Conclusion

The objective of this paper is to unify and extend results for Leontief substitution systems relevant to discrete and combinatorial research by viewing them as flows on directed hypergraphs. Sections 2-4 reinterpret some known results, and develop

new integer and binary conditions, by identifying and exploiting gainfree and disjointly reachable structures of underlying hypergraphs. Although hypergraphs in general tend to be rather abstract, we believe flows in our Leontief directed hypergraphs will prove quite intuitive for the many researchers familiar with graphs and networks. They thus make known application results easier to understand. Still, the real merit in the Leontief flow paradigm is demonstrated only by producing new application results. Section 5.5's extended bilinear characterization of languages in $\mathscr{P}$ is one, but we believe there are many more to follow. Thus we advocate the search for Leontief substitution flow applications conforming to our gainfree and disjointly reachable properties as a worthy direction for future of research.

Acknowledgement

The first author of this paper, Professor Robert (Bob) G. Jeroslow, died unexpectedly in August 1988 while attending the Mathematical Programming Symposium. At the time of his death this paper was only partially complete, so the other three authors are responsible for its present form. However, like all researchers fortunate enough to have worked with Bob we have benefited enormously from his insights and genuine warmth — may his memory be blessed.

We also thank the referees for pointing out several errors in an original draft of the paper, and for making many helpful suggestions.

References

I. Adler and S. Cosares, "Strongly polynomial algorithms for linear programming problems with special structure," Working Paper, Department of IEOR, University of California (Berkeley, CA) and Bell Communications Research (Piscataway, NJ, 1989).

M.W. Bern, E.L. Lawler and A.L. Wong, "Linear time computation of optimal subgraphs of decomposable graphs," Working Paper, Computer Science Division, University of California (Berkeley, CA, 1985).

B.A. Campbell, "Steiner tree problems on special planar graphs," Ph.D. dissertation, Department of Industrial Engineering, Purdue University (West Lafayette, IN, 1987).

V. Chandru and J.N. Hooker, "Logical inference: A mathematical programming perspective," Working Paper, CC-88-24, Purdue University (West Lafayette, IN, 1988).

A. Charnes and W.M. Raike, "One-pass algorithms for some generalized network problems," *Operations Research* 14 (1966) 914-924.

S. Cosares and I. Adler, "Advantageous properties of dual transshipment polyhedra," Working Paper, Department of Industrial Engineering and Operations Research, University of California (Berkeley, CA, 1987).

G.B. Dantzig, "Optimal solution of a dynamic Leontief model with substitution," *Econometrica* 23 (1955) 295-302.

D. Dobkin, R.J. Lipton and S. Reiss, "Linear programming is log-space hard for $\mathscr{P}$," *Information Processing Letters* 8 (1979) 96-97.

W.F. Dowling and J.H. Gallier, "Linear time algorithms for testing the satisfiability of Horn formulae," *Journal of Logic Programming* 1 (1984) 267-284.

J. Edmonds and R. Giles, "A min-max relation of submodular functions on graphs," in: P.L. Hammer, et al., eds. *Studies in Integer Programming, Annals of Discrete Mathematics* 1 (1977) 185-204.

G.D. Eppen and R.K. Martin, "Solving multi-item capacitated lot-sizing problems using variable redefinition," *Operations Research* 35 (1987) 832–848.

R.E. Erickson, "Minimum-concave-cost single-source network flows," Ph.D. dissertation, Department of Operations Research, Stanford University (Stanford, CA, 1978).

R.E. Erickson, "Optimality of stationary halting policies and finite termination of successive approximations," *Mathematics of Operations Research* 13 (1988) 90–98.

R.E. Erickson, C.L. Monma and A.F. Veinott, Jr., "Send-and-split method for minimum-concave-cost network flows", *Mathematics of Operations Research* 12 (1987) 634–664.

F.R. Giles and W.R. Pulleyblank, "Total dual integrality and integer polyhedra," *Linear Algebra and its Applications* 25 (1975) 191–196.

R.C. Grinold, "The Hirsch conjecture in Leontief substitution systems," *SIAM Journal on Applied Mathematics* 21 (1971) 483–485.

R.A. Howard, *Dynamic Programming and Markov Processes* (MIT Press, Cambridge, MA, 1960).

R.G. Jeroslow, "Computation-oriented reductions of predicate to propositional logic," *Decision Support Systems* 4 (1988) 183–197.

R.G. Jeroslow and J. Wang, "Dynamic programming, integral polyhedra, and Horn clause knowledge bases," *ORSA Journal on Computing* 1 (1989) 7–19.

N.D. Jones and W.T. Laaser, "Complete problems for deterministic polynomial time," *Proceedings of Sixth Annual ACM Symposium on Theory of Computing*, Seattle, WA, April 30–May 2, 1974, pp. 40–46.

J.G. Kemeny and J.L. Snell, *Finite Markov Chains* (Van Nostrand, Princeton, NJ, 1960).

L.G. Khachian, "A polynomial algorithm in linear programming," *Soviet Mathematics Doklady* 20 (1979) 191–194.

G.J. Koehler, A.B. Whinston and G.P. Wright, *Optimization Over Leontief Substitution Systems* (North-Holland and American Elsevier, Amsterdam, New York, 1975).

W.W. Leontief, *Structure of the American Economy, 1919–1939* (Oxford University Press, New York, 1951, 2nd ed.).

R.K. Martin, "Generating alternative mixed-integer programming models using variable redefinition," *Operations Research* 35 (1987) 820–831.

R.K. Martin, R.L. Rardin and B.A. Campbell, "Polyhedral characterization of discrete dynamic programming," *Operations Research* 38 (1990) 127–138.

U.G. Rothblum and P. Whittle, "Growth optimality for branching Markov decision chains," *Mathematics of Operations Research* 7 (1982) 582–601.

A. Schrijver, *Theory of Linear and Integer Programming* (Wiley, New York, 1986).

J.D. Ullman and A. Van Gelder, "Efficient test for top-down termination of logical rules," *Journal of the Association for Computing Machinery* 35 (1988) 345–373.

A.F. Veinott, Jr., "Extreme points of Leontief substitution systems," *Linear Algebra and its Applications* 1 (1968) 181–194.

A.F. Veinott, Jr., "Minimum concave-cost solution of Leontief substitution models of multi-facility inventory systems," *Operations Research* 17 (1969a) 262–291.

A.F. Veinott, Jr., "Discrete dynamic programming with sensitive discount optimality criteria," *Annals of Mathematical Statistics* 40 (1969b) 1635–1660.

H.M. Wagner, "On a class capacitated transportation problems," *Management Science* 5 (1959) 304–318.

A. Walker, ed., M. McCord, J.F. Sowa and W.G. Wilson, *Knowledge, Systems and Prolog* (Addison-Wesley, Reading, MA, 1987).

T.V. Wimer, S.T. Hedetniemi and R. Laskar, "A methodology for constructing linear graph algorithms," *Congressus Numerantium* 50 (1985) 43–60.