Reconciling Financial Information at Varied Levels of Aggregation*

ANIL ARYA, *The Ohio State University*

JOHN C. FELLINGHAM, *The Ohio State University*

BRIAN MITTENDORF, *Yale School of Management*

DOUGLAS A. SCHROEDER, *The Ohio State University*

Abstract

Financial statements summarize a firm's fiscal position using only a limited number of accounts. Readers often interpret financial statements in conjunction with other information, some of which may be aggregated in a different way (or not at all). This paper exploits properties of the double-entry accounting system to provide a systematic approach to reconciling diverse financial data. The key is the ability to represent the double-entry system by network flows and, thereby, access well-recognized network optimization techniques. Two specific uses are investigated: the reconciliation of audit evidence with management-prepared financial statements, and the creation of transaction-level financial ratios.

Keywords Aggregation; Auditing; Double entry; Ratio analysis

JEL Descriptors C61, M41, M42

Le rapprochement de l'information financière à divers niveaux d'agrégation

Condensé

Les utilisateurs des états financiers reçoivent des informations dont le niveau d'agrégation varie. Les rapports comptables contiennent des données relatives aux flux périodiques, dans l'état des résultats, et des données qui résultent de cumuls dans le temps (reflétant la situation financière), dans le bilan. De plus, l'information comptable que renferment les états financiers est elle-même l'expression synthétique de nombreuses opérations sous-jacentes. Et le processus ne s'arrête pas là puisque les ratios touchant les principales activités représentent encore un autre échelon d'agrégation.

* Accepted by Steve Huddart. We thank Rick Antle, Joel Demski, Ron Dye, Jon Glover, Karl Hackenbrack, Steve Huddart, Yuji Ijiri, Pierre Liang, Paul Newman, Eric Spires, Shyam Sunder, Rick Young, David Ziebart, and especially an anonymous referee, and workshop participants at Carnegie Mellon, Florida, North Carolina, Ohio State, and Yale for helpful comments. Anil Arya acknowledges financial assistance from the John J. Gerlach Chair. John Fellingham acknowledges financial assistance from the H. P. Wolfe Chair.

Dans le premier cas, il s'agit de déterminer si les états financiers, ou les contraintes de demande, coïncident avec les éléments probants, ou les contraintes de capacité. En d'autres termes, les auteurs se demandent s'il existe un ensemble d'opérations qui satisfont les deux ensembles de contraintes. Une variante de l'algorithme standard du flot maximum et de la coupe minimum permet de traiter cette question de manière efficace et même d'obtenir une réponse concluante. S'il existe une solution réalisable au problème de vérification, l'algorithme de flot maximum et de coupe minimum produit un vecteur d'opérations conforme tant aux éléments probants qu'aux états financiers. Dans le cas contraire, l'algorithme révèle un hyperplan séparateur — qui établit la distinction entre les états financiers et l'espace exploré grâce à la pondération des colonnes de la matrice à partie double, les pondérations étant limitées par les éléments probants.

Dans le second cas, les auteurs vont au-delà du contrôle de cohérence et s'intéressent à l'optimisation. En supposant que les informations dont on dispose relativement au solde de compte et à l'opération soient cohérentes, que peut-on dire au sujet des autres données financières à l'égard desquelles on ne dispose guère d'information directe ? Les auteurs se penchent plus particulièrement sur le calcul des limites relatives aux ratios financiers fondés sur les opérations. Ils utilisent pour ce faire une proche variante d'une autre technique de réseau bien connue, l'algorithme du plus court chemin par ajustement successif.

Une façon systématique d'aborder le problème des ratios financiers peut contribuer à corriger les inexactitudes des ratios financiers traditionnels. Ainsi, le calcul des ratios de rotation des comptes clients et des comptes fournisseurs exige la connaissance des ventes et des achats à crédit. Ces données liées aux opérations sont habituellement obtenues par approximation : les ventes totales sont substituées aux ventes à crédit et les achats à crédit sont remplacés par la variation des stocks majorée du coût des marchandises vendues. Plutôt que d'utiliser des approximations imparfaites, il se peut que l'on préfère calculer les limites possibles des ratios, ce qui permettrait en outre d'amalgamer aisément au calcul d'autres informations.

Les problèmes précis sur lesquels se penchent les auteurs mettent en relief un thème sous-jacent : la tenue des comptes en partie double suppose un réseau de comptes reliés entre eux qui nécessitent une vision holistique de l'information financière. Même lorsque les opérations sont indépendantes, le simple fait de débiter et de créditer simultanément des comptes engendre une corrélation dans les soldes des comptes. Pour reprendre l'analogie du casse-tête, c'est cette corrélation qui conduit à la concordance des éléments d'information et à l'apprentissage indirect. Heureusement, les diagrammes de réseau et les techniques d'optimisation de réseau facilitent la compréhension des liens que renferme le système comptable, et les progiciels commerciaux contenant des sous-programmes d'optimisation de réseau préétablis simplifient la résolution des problèmes comptables d'envergure raisonnable.

1. Introduction

Financial statement users encounter information at varied levels of aggregation. Accounting reports include periodic flow numbers in the income statement and time-aggregated stock (financial status) numbers in the balance sheet. Further, the account information in the statements is itself a summary of a large number of

underlying transactions. Even this is not the end of the aggregation process. Ratios representing key activities embed yet another layer of summary.

When information is presented at varied levels of aggregation, individual pieces of data are important, not just for new information they directly provide, but also for indirect information cascades they potentially create. A rough analogy can be made with a jigsaw puzzle, where correctly placing a single piece can reveal how several other pieces fit. Importantly, correctly placing a piece can also reveal some incorrectly placed pieces. In a similar fashion, we look to see whether or not information at varied levels of aggregation fits together. We view this as a reconciliation or a consistency check exercise. If the data are consistent, we then ask whether information at different aggregation levels can be combined in a meaningful (useful) manner.

More precisely, we model aggregation via double entry. The reasons for focusing on this particular linear aggregation rule are obvious.[1] While firm activities and reporting complexities have varied over the years, the underlying double-entry mechanics have endured. In order to obtain a better appreciation for the unique properties of the double-entry system, we address two questions. First, is a given set of financial data concerning both transactions and account balances mutually consistent? Second, if so, what other information can be gleaned by joint consideration of the transactions and accounts data? The first question is couched in terms of an auditing problem: is gathered audit evidence consistent with the financial statements presented by a client? The second question is couched in terms of a ratio analysis exercise: given what is known about a firm from its financial statements and other disclosures, what can be said about some of its financial ratios?

The role of double entry in our inquiry is unmistakable. Double entry allows an analogue to well-known network optimization techniques. Networks are characterized by directed paths connecting various locations (nodes).[2] The paths are subject to capacity constraints, the nodes are subject to demand constraints, and travel (or shipment) along each path may be associated with a specific cost. Such networks have been used to answer a myriad of questions about such issues as efficient shipping routes, task assignments, machine schedules, and utility delivery plans.

The connection between networks and double-entry bookkeeping comes from viewing accounts as nodes and transactions as paths connecting nodes. That is, a transaction consists of debiting one account, or node, and crediting another. In this case, the network demand constraints are simply the (change in) financial statement balances, or account debits less credits. Incorporating information at the transaction level is also straightforward. Transactions-related data place capacity constraints on the associated path. With the network optimization connection in hand, answering the two questions we raised above turns out to be rather straightforward.

The first question asks whether the financial statements, or demand constraints, are consistent with the audit evidence, or capacity constraints. In other words, does there exist a set of transactions that satisfies both sets of constraints? A variant of the standard max flow–min cut algorithm addresses the question efficiently. Further, the algorithm provides a conclusive answer. If the audit problem

has a feasible solution, the max flow–min cut algorithm yields a transactions vector that is consistent with both the evidence and the financial statements. Otherwise, the algorithm reveals a separating hyperplane — the hyperplane that separates the financial statements from the space spanned by weighting the columns of the double-entry matrix, where the weights are bounded by the audit evidence.

The second problem we address goes beyond a consistency question to one of optimization. That is, assuming that the account balance and transaction information at one's disposal are consistent, what can be said about other financial data for which there is little or no direct information? Specifically, we look at the calculation of bounds on transactions-based financial ratios. In this case, a slight variation of another well-known network technique, the successive shortest-path algorithm, proves helpful.[3]

A systematic approach to the financial ratio problem may help fix inaccuracies in traditional financial ratios. For example, credit sales and credit purchases are needed to compute the receivables turnover and payables turnover ratios. These transaction amounts are typically approximated: credit sales is replaced by total sales and credit purchases by change in inventory plus cost of goods sold. Instead of crude approximations, one may prefer to calculate feasible bounds on a ratio. Such an approach also allows other information to be easily assimilated into the calculation.

The specific problems we address underscore an underlying theme — double entry introduces an interconnected network of accounts which necessitates a holistic view of financial information. Even when transactions are independent, the mere act of simultaneously debiting and crediting an account introduces a correlation in account balances (Ijiri 1975; Sunder 1997). Returning to our jigsaw analogy, it is this correlation that leads to interlocking pieces of information and indirect learning. Fortunately, network diagrams and network optimization techniques simplify the process of understanding linkages in the accounting system. And the availability of commercial software packages with preprogrammed network optimization routines facilitates solving realistically sized accounting problems.

2. Representation of the double-entry system

Financial statements linearly aggregate information in several transactions using only a few account balances. Let $\boldsymbol{A}$ denote this transformation matrix. $\boldsymbol{A}$ has m rows and n columns, where m is the number of accounts and n is the number of transactions, $m < n$. There are two nonzero entries in each column corresponding to the accounts that are connected by a journal entry. We adopt the following sign convention for the nonzero entries: the account that is debited is assigned $+1$ and the account that is credited is assigned -1. For obvious reasons, we refer to $\boldsymbol{A}$ as the double-entry matrix. The financial statement preparation process is represented mathematically by $\boldsymbol{Ay} = \boldsymbol{x}$, where $\boldsymbol{y}$ is an n-length vector of transaction amounts and $\boldsymbol{x}$ is the resulting m-length vector of (changes in) account balances.[4]

The (change in) balance in account i is denoted x_i, $i = 1, 2, \ldots, m$. Also, the transaction amount corresponding to the journal entry that credits account i and debits account j is denoted y_{ij}.

A basic premise of the paper is that a reader of financial statements observes $\boldsymbol{x}$ but not $\boldsymbol{y}$. This is not to say that the reader may not have some other sources of information about $\boldsymbol{y}$ (for example, footnote disclosures, acquired expertise about the firm, or other gathered evidence). While interpreting the financial statements in light of this other information, the reader can take advantage of the fact that the aggregation process is double entry.

Double entry allows a directed graph representation of the accounting system in which nodes correspond to accounts and edges correspond to journal entries (Arya, Fellingham, Glover, Schroeder, and Strang 2000). The graph is directed in that the arrow of each edge points to the account that is debited by that journal entry. As we will shortly see, it is this representation that opens the door to exploiting network flow algorithms. Before turning to such an analysis, we present parameters for an example that we will use throughout the paper to provide intuition.

Consider the following financial statements:

Balance sheet	*Ending balance*	*Beginning balance*	**Income statement**	
Cash	11	8	Sales	7
Receivables	8	7	CGS	3
Inventory	3	4	G&A	3
Plant	11	10	Income	1
Total Assets	33	29		
Payables	10	7		
Owners' equity	23	22		
Total liability and equity	33	29		

Though the specific transaction amounts are unknown, the reader of the financial statements knows about the firm's main operations and, hence, the transactions it could undertake in the normal course of business. These transactions, and the matching journal entries, are listed below. The journal entries involve Cash, Receivables, Inventory, Plant, Payables, Sales, Cost of Goods Sold (CGS), and General and Administrative (G&A). These accounts are labeled 1–8, respectively.

Activity	**Credit account**	**Debit account**	**Amount**
Collection of receivables	Receivables (#2)	Cash (#1)	y_{21}
Cash purchase of plant	Cash (#1)	Plant (#4)	y_{14}
Payment of payables	Cash (#1)	Payables (#5)	y_{15}
Bad debt expense	Receivables (#2)	G&A (#8)	y_{28}
Credit sales	Sales (#6)	Receivables (#2)	y_{62}
Depreciation, period cost	Plant (#4)	G&A (#8)	y_{48}
Recognition of CGS	Inventory (#3)	CGS (#7)	y_{37}
Accrued expenses	Payables (#5)	G&A (#8)	y_{58}
Purchase inventory on credit	Payables (#5)	Inventory (#3)	y_{53}
Depreciation, product cost	Plant (#4)	Inventory (#3)	y_{43}

The A matrix representing the firm's journal entries and the x vector representing the financial statement account balances are presented below.

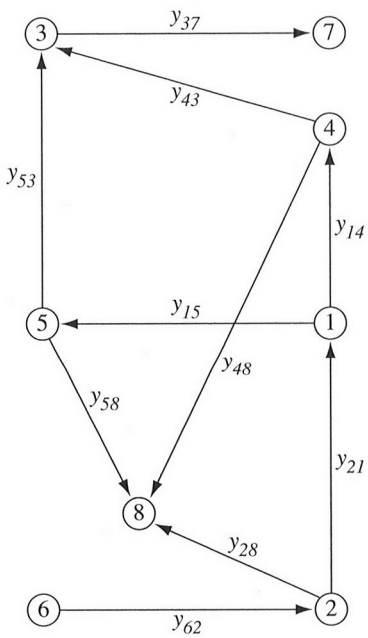
x		A									
3	Cash	1	−1	−1	0	0	0	0	0	0	0
1	Receivables	−1	0	0	−1	1	0	0	0	0	0
−1	Inventory	0	0	0	0	0	0	−1	0	1	1
1	Plant	0	1	0	0	0	−1	0	0	0	−1
−3	Payables	0	0	1	0	0	0	0	−1	−1	0
−7	Sales	0	0	0	0	−1	0	0	0	0	0
3	CGS	0	0	0	0	0	0	1	0	0	0
3	G&A	0	0	0	1	0	1	0	1	0	0

The equivalent directed graph for A in our example is presented in Figure 1.

3. Consistency of information

Reconciling transactions-based and accounts-based information involves checking whether they are mutually consistent. A setting in which this issue is at the forefront is auditing.[5] While an auditor does not express an opinion regarding the

Figure 1 The directed graph representation of A for the example



underlying transactions, "a principal means of establishing the validity of a balance sheet and income statement is to trace the statement figures to the accounting records and back through the records to the original evidence of transactions" (Pany and Whittington 1994, 40). The emphasis on gathering transactions-related evidence is most apparent in the cycles approach (Arens and Loebbecke 2000).[6] Auditing the sales and collection cycle, for example, entails checking credit sales, sales returns, cash collections, and charge-offs to confirm the balance in accounts receivable.

To elaborate, an auditor gathers evidence regarding both account balances and transactions. Accounts-related evidence can, of course, be directly compared with the account balances presented by management. In this section, we focus on incorporating transactions-related evidence in the verification exercise. Audit evidence limits permissible transaction amounts. Let vectors $\boldsymbol{y}_L$ and $\boldsymbol{y}_U$ denote the lower and upper bounds imposed by audit evidence on the n transaction amounts. These bounds may reflect the fact that the auditor employs interval estimates (confidence intervals) when evidence is gathered through sampling techniques. Alternatively, the bounds can reflect tolerable misstatements or represent bounds of reasonable industry benchmarks in the analytical review phase. (Precise evidence in the form of point estimates is a special case of this framework.)

The auditor's problem is to determine whether or not there exist transaction amounts that are consistent with management prepared financial statements ($\boldsymbol{Ay} = \boldsymbol{x}$) and audit evidence ($\boldsymbol{y}_L \leq \boldsymbol{y} \leq \boldsymbol{y}_U$). Denoting $\boldsymbol{y} - \boldsymbol{y}_L$ by $\boldsymbol{y}'$, this reconciliation problem is equivalent to finding a $\boldsymbol{y}'$ that satisfies $\boldsymbol{Ay}' = \boldsymbol{x}'$ and $\boldsymbol{0} \leq \boldsymbol{y}' \leq \boldsymbol{y}_U - \boldsymbol{y}_L$, where $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{Ay}_L$. Assume, for simplicity, that $\boldsymbol{y}$ is not bounded from above (that is, $\boldsymbol{y}_U = \infty$).[7] Then, the reconciliation problem reduces to finding a $\boldsymbol{y}'$ that solves $\boldsymbol{Ay}' = \boldsymbol{x}'$ and $\boldsymbol{y}' \geq \boldsymbol{0}$. Because any reconciliation exercise can always be rewritten in this form (and upper bounds can be incorporated), there is no loss of generality in studying the problem with $\boldsymbol{y}_L = \boldsymbol{0}$ and $\boldsymbol{y}_U = \infty$, as we do from here on.

Below, we present an algorithm to solve the reconciliation problem. At the heart of the algorithm is the max flow–min cut method. The max flow–min cut method is an efficient graphical approach for solving network flow problems.[8] The method can be viewed as a simplification of the long-standing linear programming "simplex" technique, where the simplification arises because network flow problems can be represented by an incidence matrix. An incidence matrix is one in which each column contains a single +1 and a single −1, and all remaining elements are 0 (e.g., Strang 1988, 1027). The double-entry matrix $\boldsymbol{A}$ is clearly an incidence matrix and, hence, permits the following algorithm to determine consistency (see Ahuja, Magnanti, and Orlin 1993, 166–206, and in particular 169–70).[9]

OBSERVATION 1. *The following consistency algorithm (an application of the max flow–min cut algorithm) reconciles transactions-based evidence with account balance information by yielding either a consistent transactions vector or a separating hyperplane.*

(1) Construct graph G with node set V (vertices) and arc set E (edges) as follows:

(a) Add two "artificial" nodes, a source node, s, and a sink node, t, to the directed graph representation.

(b) Connect s to all accounts with a credit balance, and connect all accounts with a debit balance to t.

(2) Select an initial flow in G — say, $y_{ij} = 0$ for all $(i, j) \in E$.

(3) Specify the capacity (maximum permissible flow) of each arc (i, j) by u_{ij}. u_{ij} is ∞ for arcs not involving s or t. (If $\boldsymbol{y_U}$ placed a nontrivial upper bound on some transaction amount, then the capacity of the corresponding arc would reflect the bound.) For arcs involving s or t, the capacity is the absolute value of the associated node's balance, or $u_{sj} = |x_j|$, and $u_{it} = |x_i|$.

(4) Construct a new graph, G', as follows:

(a) G' has the same nodes as G.

(b) If $y_{ij} < u_{ij}$ in G, place arc (i, j) in G'. The capacity of this arc in G' is the remaining capacity, $u_{ij} - y_{ij}$.

(c) If $y_{ij} > 0$ in G, place the reverse arc (j, i) in G'. This arc's capacity in G' is y_{ij}.

(5) In G', find a (directed) path P from s to t. Determine the maximum amount Δ that can flow along the path (that is, the minimum capacity along the path).

(6) In G, add Δ to y_{ij} if (i, j) is in P, subtract Δ from y_{ij} if (j, i) is in P, and leave y_{ij} unchanged otherwise.

(7) Repeat steps (4), (5), and (6) until there are no paths left in G' flowing from the source node to the sink node.

(8) If all the arcs in G connected to the source and sink nodes are saturated (that is, $y_{sj} = u_{sj}$ and $y_{it} = u_{it}$), the financial statements and audit evidence are consistent. In this case, the y_{ij} in G is a consistent transactions vector.

(9) If all arcs in G connected to the source and sink nodes are not saturated, the financial statements and audit evidence are inconsistent. In this case, construct a vector $\boldsymbol{\lambda}$ with m positions, one for each node in the audit problem network other than s and t (that is, one for each account). For each node in G' that can be reached from s, assign a one to the corresponding position in $\boldsymbol{\lambda}$. Assign a zero to all other positions in $\boldsymbol{\lambda}$. The resulting m-length vector $\boldsymbol{\lambda}$ confirms inconsistency as it identifies a separating hyperplane.

To see the algorithm at work, return to the example. Figure 2 shows the initial G and G'. The graphs represent the original directed graph from Figure 1 with

added arcs and nodes labeled with dashes. For simplicity, the uncapacitated arcs ($u_{ij} = \infty$) are left unlabeled in G'.

Note that G' in Figure 2 has 10 possible paths from s to t. Step (5) allows any of those paths to be chosen. And, as the algorithm unfolds, some of those paths will be shut off as certain arcs become saturated. Hence, the algorithm may take fewer than 10 iterations and there may be multiple solutions to the algorithm at the end of step (7). Fortunately, each possible solution will lead to the appropriate conclusion in steps (8) and (9).

For this example, one implementation of the algorithm is shown below, as identified by path P and flow Δ:

Iteration 1: $P = \{(s, 6), (6, 2), (2, t)\}, \Delta = 1$
Iteration 2: $P = \{(s, 6), (6, 2), (2, 1), (1, t)\}, \Delta = 3$
Iteration 3: $P = \{(s, 6), (6, 2), (2, 1), (1, 4), (4, t)\}, \Delta = 1$
Iteration 4: $P = \{(s, 6), (6, 2), (2, 1), (1, 4), (4, 3), (3, 7), (7, t)\}, \Delta = 2$
Iteration 5: $P = \{(s, 5), (5, 8), (8, t)\}, \Delta = 3$
Iteration 6: $P = \{(s, 3), (3, 7), (7, t)\}, \Delta = 1$

At the end of these iterations, G and G' are as in Figure 3.

In Figure 3, there are no paths in G' following the direction of arrows that emanate from the source node and finish in the sink node. Hence, we are at a stopping point. To see whether step (8) or (9) applies, we check for saturation. Since in G all arcs from the source and to the sink are saturated (in G', there are no paths departing s and no paths arriving at t), the transaction evidence and account balance information are consistent. A consistent transactions vector immediately follows from G: $\boldsymbol{y} = (y_{21}, y_{14}, y_{15}, y_{28}, y_{62}, y_{48}, y_{37}, y_{58}, y_{53}, y_{43}) = (6, 3, 0, 0, 7, 0, 3, 3, 0, 2)$. The reader can verify that $\boldsymbol{Ay}$ is indeed $\boldsymbol{x}$ in the example.

Now, let us consider a slight change to the example that results in a conclusion that the transaction evidence is inconsistent with the account balance information. Assume $\boldsymbol{x} = (2, 1, -1, 1, -2, -3, 1, 1)$. With this $\boldsymbol{x}$, the algorithm yields the following iterations:

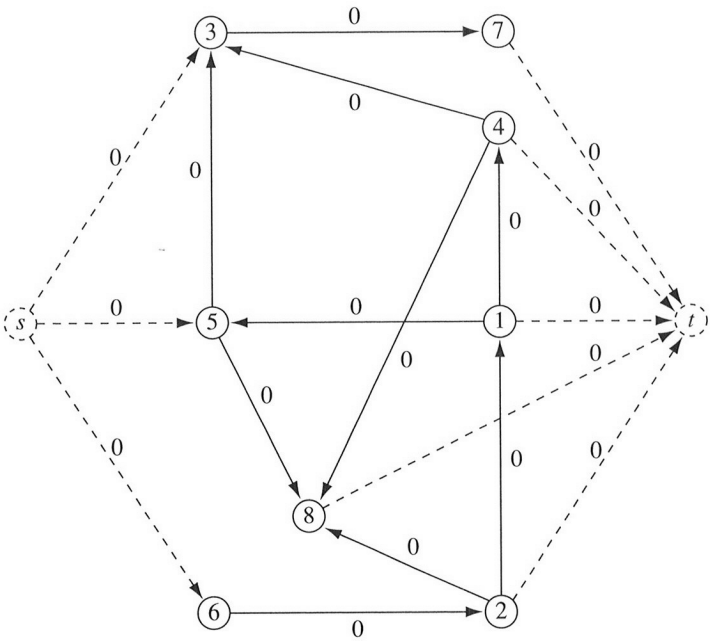
Iteration 1: $P = \{(s, 6), (6, 2), (2, t)\}, \Delta = 1$
Iteration 2: $P = \{(s, 6), (6, 2), (2, 1), (1, t)\}, \Delta = 2$
Iteration 3: $P = \{(s, 5), (5, 8), (8, t)\}, \Delta = 1$
Iteration 4: $P = \{(s, 3), (3, 7), (7, t)\}, \Delta = 1$

At the end of these iterations, G and G' are as in Figure 4.

From Figure 4, there are no paths left in G' that start at the source node and finish at the sink node. Hence, we are at a stopping point. But, in G, the arcs $(s, 5)$ and $(4, t)$ are not saturated, so there is an inconsistency. The nodes that can be reached from s in G' are 3, 5, 7, and 8, so $\boldsymbol{\lambda} = (0, 0, 1, 0, 1, 0, 1, 1)$.

Figure 2 Initial G with flows and G' with capacities for $\boldsymbol{x} = (3, 1, -1, 1, -3, -7, 3, 3)$

Initial G



Initial G'

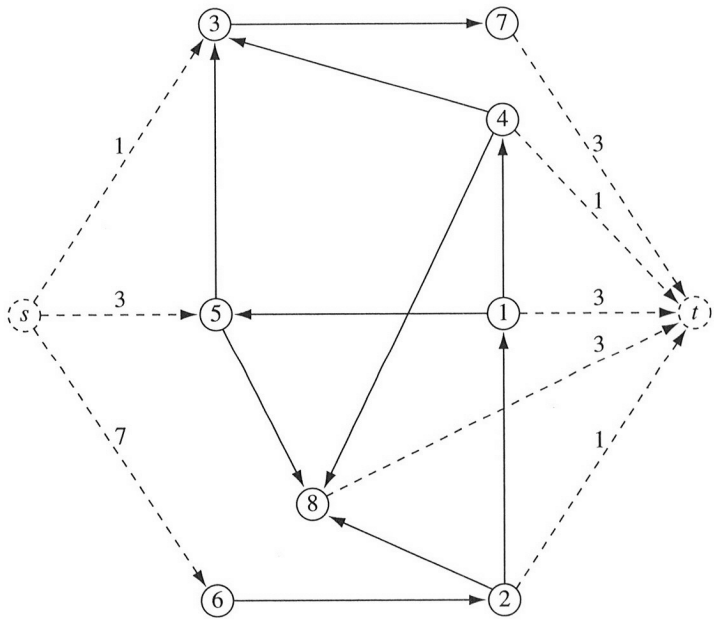
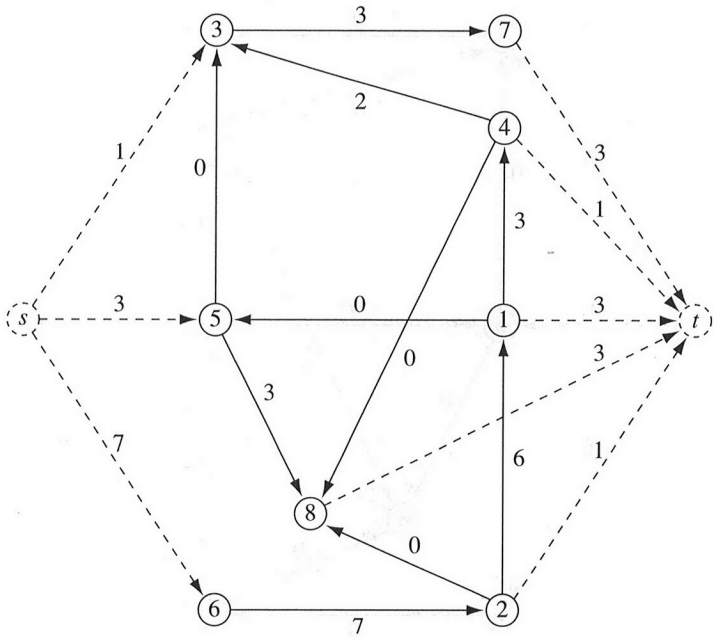


Figure 3 Final G with flows and G' with capacities for $\boldsymbol{x} = (3, 1, -1, 1, -3, -7, 3, 3)$

Final G



Final G'

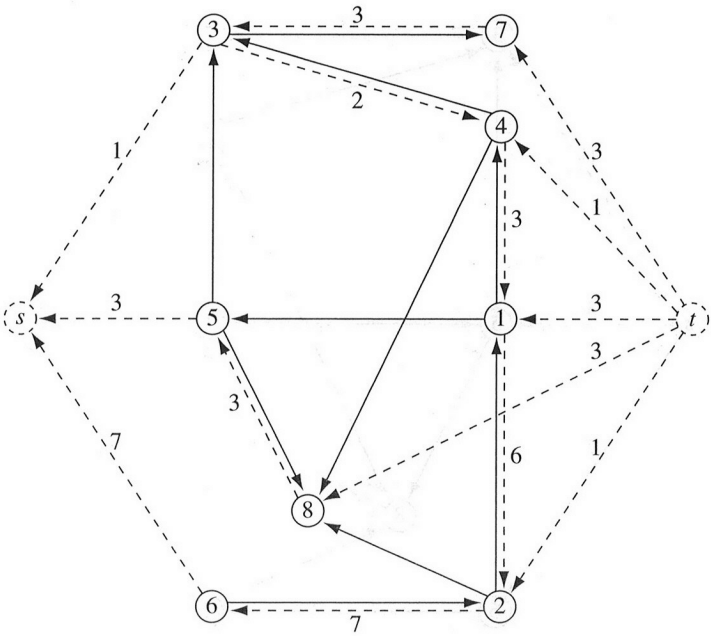
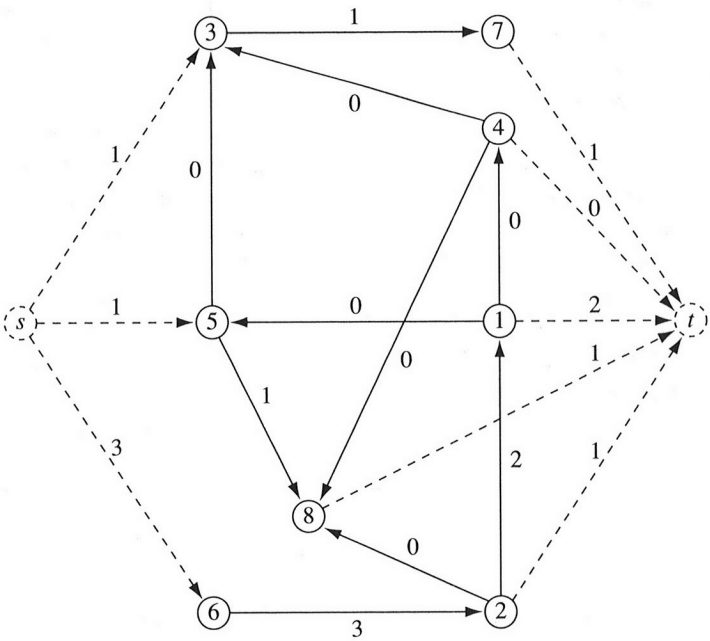
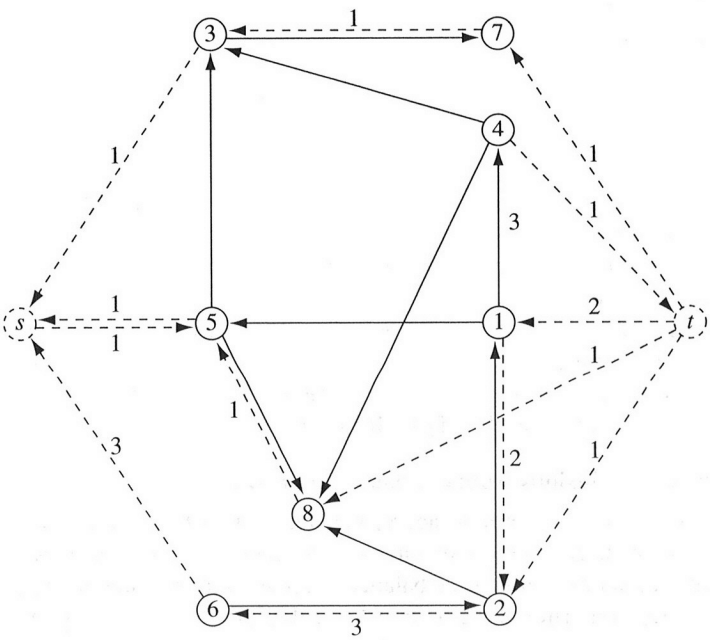


Figure 4 Final G with flows and G' with capacities for $\boldsymbol{x} = (2, 1, -1, 1, -2, -3, 1, 1)$

Final G



Final G'



This $\boldsymbol{\lambda}$, coupled with the theorem of the separating hyperplane, confirms there is an inconsistency. The theorem of the separating hyperplane (see Strang 1988, 419–20) states that either there is a non-negative solution to $\boldsymbol{Ay} = \boldsymbol{x}$ or there exists a $\boldsymbol{\lambda}$ such that $\boldsymbol{\lambda}^T\boldsymbol{x} < 0$ and $\boldsymbol{A}^T\boldsymbol{\lambda} \geq \boldsymbol{0}$. For the example, $\boldsymbol{\lambda}^T\boldsymbol{x} = -1 < 0$ and $\boldsymbol{A}^T\boldsymbol{\lambda} = (0, 0, 1, 1, 0, 1, 0, 0, 0, 1) \geq \boldsymbol{0}$.

In geometric terms, the $\boldsymbol{\lambda}$ vector in the theorem essentially identifies the hyperplane that separates $\boldsymbol{x}$ from the space spanned by placing non-negative weights on the columns of $\boldsymbol{A}$ (also known as the cone of $\boldsymbol{A}$). In particular, the separating hyperplane is orthogonal to the $\boldsymbol{\lambda}$ vector. Because our running example is in 8-space (involves 8 accounts), visualizing the hyperplane and, hence, appreciating the theorem is not an easy task. Fortunately, the double-entry system with its zero/one characterization of the hyperplane lends itself to an intuitive schema.

$\boldsymbol{\lambda}$ divides the accounts in two account groups (those labeled with a zero and those labeled with a one). Figure 5 lists the groups along with the sum of balances in each group for the example.

The figure also lists all transactions that link an account in any group with an account in the other group. In the example, all such transactions point to the one-account group (involve debits to the one-accounts). How, then, can the one-account group have a net credit balance? It is this inconsistency that proves the financial statements must have been generated using another transaction — in particular, a transaction involving a credit to a one-account and a debit to a zero-account.

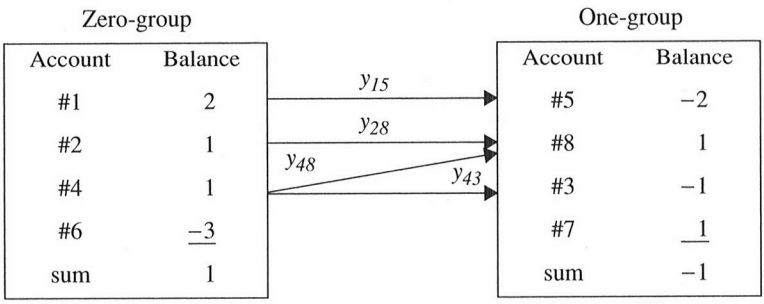
The two conditions in the separating hyperplane conduct precisely the exercise in Figure 5. $\boldsymbol{A}^T\boldsymbol{\lambda} \geq \boldsymbol{0}$ means the only permissible transactions connecting the two account groups involve credits to a zero-account and debits to a one-account. $\boldsymbol{\lambda}^T\boldsymbol{x} < 0$ indicates that the financial statements specify a net credit balance in the one-account group.

Besides providing the separating hyperplane that confirms an inconsistency, the algorithm may also provide some evidence about the root of the problem. For example, suppose an auditor determines audit evidence and financial statements are inconsistent and suspects a firm may be incorrectly capitalizing an expense (say, research and development [R&D]). Before investigating such suspicions further, the auditor may wish to revisit the consistency question — that is, add the extra transaction of incorrectly classifying R&D (debit the appropriate asset and credit the appropriate expense). With this additional transaction in play, does the inconsistency remain? If there remains an inconsistency, one can be sure that misclassification of R&D is not the sole reason the financial statements and evidence cannot be reconciled. If the additional transaction removes the inconsistency, suspicions that the firm is misclassifying R&D are bolstered.

4. Drawing conclusions from consistent information

Although the accounting process aggregates transactions by a few account balances, this often is not the end of the summarization process. For example, users of financial statements commonly summarize balance sheet and income statement numbers by meaningfully constructed ratios that are calculated for a variety of purposes, including analytical review. The traditional ratios make use of account balance

Figure 5 Representation of the separating hyperplane theorem for $x = (2, 1, -1, 1, -2, -3, 1, 1)$



information because that is the information directly available to users. Even if a ratio depends on transaction information, it is estimated by approximating the transaction amount using financial statement balances. Turnover ratios are a case in point where, for example, credit sales in the receivables turnover ratio is approximated by total sales. The previous section suggests an alternative: the user can calculate the range of feasible values for the ratio rather than approximate the transaction component of the ratio using account balances.

The calculation of ratio bounds necessitates solving an optimization problem. Specifically, we minimize a weighted linear combination of transaction amounts while ensuring that transaction amounts are consistent with the financial statements and with any gathered evidence. (Finding the maximum of a weighted combination can also be couched as a minimization problem by reversing the sign on each weight.) Formally, the problem is:

$$\text{Min } \boldsymbol{w}^T\boldsymbol{y}$$

s.t.

$$\boldsymbol{A}\boldsymbol{y} = \boldsymbol{x}$$

$$\boldsymbol{y} \geq \boldsymbol{0}$$

In the statement of the problem, w_{ij} is the weight associated with transaction y_{ij}. The weights are determined by the transaction (or set of transactions) for which a range of values is sought. For example, if the lower bound on $y_{ij} - y_{km}$ is sought, then $w_{ij} = 1$, $w_{km} = -1$, and all other weights are zero. The optimality question can be addressed with only slight modifications to the consistency algorithm. The optimality algorithm follows two stages (see, e.g., Ahuja et al. 1993, 320–4; Bazaraa, Jarvis, and Sherali 1990, 619–20).

OBSERVATION 2. *The following two-stage optimization algorithm (a variant of the successive shortest-path algorithm) can be used to determine bounds on any linear combination of consistent transaction amounts.*

Stage 1. Checking boundedness.

From the directed graph for $\boldsymbol{A}$, identify any loop of transactions in which all arrows point in the same direction (either all clockwise or all counterclockwise). Suppose that there is such a loop. Without loss of generality, say the loop is y_{kl}, y_{lm}, and y_{mk}. If $w_{kl} + w_{lm} + w_{mk} < 0$, then the problem is unbounded — that is, the minimum is $-\infty$. If there is no such negative unidirectional loop, only then proceed to stage 2.

Stage 2. Finding a minimum.

This stage simply conducts the consistency algorithm with two changes. First, in each iteration of step (4), label each arc in G' with both its capacity and weight. The weight on arc (i, j) is w_{ij}, the weight on arcs added to the source and sink nodes (that is, (s, j) or (i, t)) is zero, and the weight on a reversed arc is the negative of the weight on the original arc (that is, $w_{ji} = -w_{ij}$). Second, in each iteration of step (5), instead of choosing any path from s to t, find a specific path, the one where the sum of arc weights in G' is the minimum (the least-weight path).

Stage 1 makes use of the notion of a transactions loop to address boundedness. The presence of such a loop implies that there is no effect on account balances when each transaction in the loop is changed by an equal amount. Clearly, if the sum of weights on the transactions in a loop is negative, there is no limit to minimization. On the other hand, if there is no such negative loop, the problem is bounded. It is only in the latter case that we move to stage 2 to find the (bounded) minimum.

Stage 2 augments the consistency algorithm by identifying a least-weight path in each iteration. Just as the max flow–min cut algorithm was central in determining consistency, the successive shortest-path algorithm is key in determining optimality. For a simple setup with only one $w_{ij} \neq 0$, finding the least-weight path in each iteration is rather straightforward. For complex problems, however, this task could be more onerous. In such cases, procedures such as Dijkstra's shortest-path algorithm can ensure proper identification of the least-weight path (for this, and other shortest-path approaches, see Ahuja et al. 1993, chapters 4–5).

As an application of Observation 2, consider the construction of an accounts payable turnover ratio — inventory purchases on credit divided by average payables — for the paper's running example. From the financial statements, average payables equal $0.5(10 + 7) = 8.5$. Inventory purchases on credit is typically approximated by CGS plus change in inventory, which equals 2 in our example. This approximation yields a turnover ratio of $2/8.5 \approx 0.24$.

Alternatively, the optimization algorithm can be used to determine the range of values for inventory purchases on credit (y_{53}). Finding the minimum value of y_{53} can be accomplished by conducting the optimization with $w_{53} = 1$ and $w_{ij} = 0$ for the remaining transactions. Applying the algorithm to the example with the goal of minimizing y_{53} yields precisely the iterative process listed following Figure 2. Hence, Min $y_{53} = 0$. Repeating the process with $w_{53} = -1$ to determine the upper

bound reveals Max $y_{53} = 2$. Hence, the range for the accounts payable turnover ratio is $0/8.5 = 0$ to $2/8.5 \approx 0.24$. Note that the standard approximation corresponds to the maximum value the ratio can take.[10]

The range representation alerts us to the fact that the approximation used in traditional ratio analysis may be crude. In the case of the turnover ratio, the standard approximation assumed zero product cost depreciation. As a financial statement reader gathers more consistent evidence on transactions, this point estimate for the ratio is unaffected because it relies only on account balances. However, the range calculation properly incorporates the new evidence. Suppose that a careful reading of the financial statement footnotes reveals bad debt write-offs, y_{28}, were 1.5. This information alters the bounds for y_{53}: Min $y_{53} = 1.5$ and Max $y_{53} = 2$. The updated range for the accounts payable turnover ratio is $1.5/8.5 \approx 0.18$ to $2/8.5 \approx 0.24$.

Though bad debt write-offs may seem to be unrelated to accounts payable turnover, the joint consideration of financial statements and additional evidence highlights the interdependency inherent in the financial statements, one that goes unnoticed using standard ratio analysis. The interdependency is a natural consequence of double entry. The exercise of debiting an account and crediting a different account simultaneously introduces a dependency in account balances even when transactions may themselves be independent.

Taking a network approach to ratio analysis also opens doors to the creation of other ratios based on transactions. For example, instead of using times-interest-earned as a measure of ability to pay debt obligations, one may wish to calculate an interest coverage ratio of cash inflows/interest expense (here, for all transactions involving a debits to cash, $w_{ij} = 1$, and for all others, $w_{ij} = 0$). Also, to measure collection success, one may wish to compute a collections flow ratio of receivable collections/average receivables (after all, a favorable receivables turnover ratio may simply imply large bad debts).

For our example, the firm's accounts receivable turnover, sales/average receivables, is 0.93. This is not particularly encouraging. A further investigation by way of a collections flow ratio serves to heighten concerns of the firm's ability to collect from customers. The receivable collections (y_{21}) bounds are Min $y_{21} = 4$ and Max $y_{21} = 6$, yielding bounds on the collections flow ratio of 0.53 and 0.8. After including the footnote information on bad debts ($y_{28} = 1.5$), the bounds are updated to Min y_{21} = Max $y_{21} = 4.5$, yielding a collections flow number of 0.6.

There are, of course, many other conceivable transactions-based ratios that can be calculated. As with standard ratio analysis, the determination of which ratios are of use is surely a contextual exercise. The point is not that a particular ratio is of importance, but that systematically incorporating both accounts-based and transactions-based information adds a new dimension to the usual financial statement analysis toolkit.

5. Implementation issues

Though the graphical approach presented in previous sections works, it does raise concerns about applicability to realistically sized accounting systems that are presumably much larger. As it turns out, the strength of network flow algorithms is

their adaptability to large problems. In fact, it is precisely when there are a large number of nodes and arcs that the network flow algorithms really outshine standard linear programming approaches. In discussing standard network problems (in particular, shortest path, maximum flow, and minimum cost flow), Ahuja et al. (1993, 2) make this point eloquently:

> In the sense of traditional applied and pure mathematics, each of these problems is trivial to solve. It is not very difficult (but not at all obvious for the later two problems) to see that we need only consider a finite number of alternatives. So a traditional mathematician might say that the problems are well solved: Simply enumerate the set of possible solutions and choose the one that is best. Unfortunately, this approach is far from pragmatic, since the number of possible alternatives can be very large — more than the number of atoms in the universe for many practical problems! So instead, we would like to devise algorithms that are in a sense "good", that is, whose computational time is small, or at least reasonable for problems met in practice.

As alluded to previously, the reason efficient algorithms for network flows can be designed is because of the incidence property of the associated matrix. Not only does an incidence matrix have entries restricted to the values −1, 0, and 1, so do all its four fundamental subspaces, each of which can be identified graphically (see Observation 1 in Arya et al. 2000 and Strang 1988, 102). This property means any basis of the system of equations represented by an incidence matrix can be put in a triangular form. In terms of linear programming, this allows the problem to be addressed without the use of inverse operations, instead solving equations by backward substitution one variable at a time (e.g., Luenberger 1989, 124). By exploiting such features and the simple structure of the problems, network flow algorithms can improve upon the standard simplex approach.

As far as practical implementation goes, many software packages have been developed to exploit the beneficial properties of network flow algorithms (e.g., CPLEX, GENOS, LNOS, NETFLOW). Because our algorithms are an amalgam of standard network flow approaches, these packages can readily be accessed either to run the consistency algorithm or to calculate bounds on ratios. In fact, the SAS software package, with which many accountants are familiar, contains network tools as part of the SAS/OR package (for example, Proc Netdraw to draw directed graphs and Proc Netflow to implement network flow algorithms). The appendix demonstrates the use of Proc Netflow in SAS/OR for our accounting examples.

6. Conclusion

The double-entry aggregation process that is ubiquitous in financial reporting creates an interconnected network of financial accounts. The interconnectivity means some financial information that is seemingly unrelated to a financial statement reader's primary focus could prove useful. We underscore the importance of such interdependencies using two examples, one from the field of auditing (where interdependencies

play an explicit role) and one from the realm of financial statement analysis (where consideration of interdependencies is less apparent). The tasks are simplified thanks to close connections between accounting systems, network flow diagrams, and graphical optimization techniques.

Taking a network view of the accounting system can also shed light on other questions. For example, a financial statement user (or auditor) may wonder what information he should gather to have the greatest impact on his understanding of the firm. The natural tendency to seek out information about a company's core operations or about the largest (most material) transactions may turn out to be incorrect. Instead, the user may get a greater understanding by uncovering information about a relatively small transaction. Though the direct impact of such information is negligible, it could have a substantial indirect impact by narrowing the range of possible values for several other transactions. This indirect impact arises from the interconnectedness of the accounting system.

Appendix

Below, we present the SAS code and output that runs the consistency algorithm for our original example.

Program

```
title 'Audit Problem';
title2 'Path Consistency';

data acctg;
input    ffrom $ tto $ _capac_ ;
cards;
Rec         Cash        999999
Cash        Plant       999999
Cash        Payables    999999
Rec         G&A         999999
Sales       Rec         999999
Plant       G&A         999999
Inv         CGS         999999
Payables    G&A         999999
Payables    Inv         999999
Plant       Inv         999999
s           Sales       7
s           Inv         1
s           Payables    3
Cash        t           3
Rec         t           1
Plant       t           1
CGS         t           3
G&A         t           3;
```

```
proc netflow
   maxflow
   sourcenode='s' /* Quotes for case sensitivity */
   sinknode='t'
     arcdata=acctg
     arcout=path;
       tail ffrom;
       head tto;
proc print data=path; var ffrom tto _capac_ _flow_ _status_;
run;
```

Output

Audit Problem
Path Consistency

OBS	FFROM	TTO	_CAPAC_	_FLOW_	_STATUS_	
1	Inv	CGS	999999	3	KEY_ARC	BASIC
2	Rec	Cash	999999	6	KEY_ARC	BASIC
3	Rec	G&A	999999	0	LOWERBD	NONBASIC
4	Plant	G&A	999999	0	LOWERBD	NONBASIC
5	Payables	G&A	999999	3	KEY_ARC	BASIC
6	Payables	Inv	999999	2	KEY_ARC	BASIC
7	Plant	Inv	999999	0	LOWERBD	NONBASIC
8	s	Inv	1	1	UPPERBD	NONBASIC
9	Cash	Payables	999999	2	KEY_ARC	BASIC
10	s	Payables	3	3	UPPERBD	NONBASIC
11	Cash	Plant	999999	1	KEY_ARC	BASIC
12	Sales	Rec	999999	7	KEY_ARC	BASIC
13	s	Sales	7	7	UPPERBD	NONBASIC
14	Cash	t	3	3	UPPERBD	NONBASIC
15	Rec	t	1	1	UPPERBD	NONBASIC
16	Plant	t	1	1	UPPERBD	NONBASIC
17	CGS	t	3	3	UPPERBD	NONBASIC
18	G&A	t	3	3	KEY_ARC	BASIC

From log

```
NOTE: Number of nodes=10.
NOTE: Number of arcs=18.
NOTE: Number of iterations performed (neglecting any constraints)=16.
NOTE: Of these, 8 were degenerate.
NOTE: Maximal flow=11.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost=0.
NOTE: The data set WORK.PATH has 18 observations and 13 variables.
NOTE: PROCEDURE NETFLOW elapsed time was 0.73 seconds with
      13.00mb available memory.
```

The SAS output confirms consistency, because all arcs emanating from s and those linked to t are saturated (Capacity equals Flow). The same conclusion was reached when the graphical algorithm in Observation 1 was used to solve the example. The log indicates that the maximal flow is 11, which is what we had obtained earlier (the sum of Δs was 11). Further, the flows presented in the above output provide a consistent transactions vector, as specified by the flows (unrelated to s and t) in the SAS output report. Also, as mentioned previously, there is not a unique consistent transactions vector, so the graphical approach and the computer package may provide different solution vectors depending on the iterations conducted in the algorithm. Such is the case in this example.

The optimization bounds calculated in Observation 2 can be accomplished in SAS using the same code as above with one change. A cost variable (_cost_) is added to the input data, with each cost entry corresponding to the appropriate w-weight.

Endnotes

1. There is a long tradition in accounting of formally modeling the double-entry accounting system (e.g., Butterworth 1972; Ijiri 1971; Mattessich 1964; Williams and Griffin 1964). The emphasis on linear aggregation is a pervasive theme in Demski 1994 and Ijiri 1975.
2. For excellent overviews of network flow problems, see Ahuja, Magnanti, and Orlin 1993; Bazaraa, Jarvis, and Sherali 1990; and Luenberger 1989 (117–63).
3. The connection to network optimization also means that the problems can be framed as transportation problems. In this case, the graphical approach is sidestepped in favor of a set of tableaus. The tableaus require linking all accounts (nodes), not just accounts connected with sensible journal entries. This makes the transportation approach more tedious than this paper's graphical approach. We thank the referee for pointing us to the appropriate graphical algorithms.
4. $\boldsymbol{x}$ is also recorded using the same convention as in $\boldsymbol{A}$. Each entry in $\boldsymbol{x}$ equals the change in an account balance multiplied by +1, if the account is an asset or an expense (accounts with debit balance), and −1 otherwise. The beginning balance of income statement accounts is zero.
5. As a tantalizing observation, we note that an audit industry has arisen around the production of double-entry financial statements, and there appears to be no comparable audit activity in the absence of double entry.
6. Arens and Loebbecke (2000) list five basic transaction cycles: sales and collection, acquisition and payment, payroll and personnel, inventory and warehousing, and capital acquisition and repayment.
7. Incorporating a nontrivial upper bound in our setup is straightforward. The details are spelled out in Observation 1.
8. As an aside, we note that standard presentations of network algorithms make a simplifying integrality assumption — capacities, costs, and demands are integers (Ahuja et al. 1993, 6). Integrality guarantees that solutions are integers and thus simplifies the search for a solution. In our setup, where account balances are rounded up to the nearest dollar or nearest thousand, the integrality assumption is automatically

satisfied. And, financial data presented to the penny can simply be multiplied by 100 to allow for integral network flow data.

9. More specifically, the algorithm is an application of the augmenting path algorithm to a feasible flow problem (Ahuja et al. 1993, 180–1), which rests on the notion of a residual network (Ahuja et al. 1993, 177). The algorithm is easier to follow in the context of our example — the reader may wish to see Figures 2 and 3 alongside Observation 1.
10. One may argue that a more appropriate definition of the turnover ratio in our setting would include all credit purchases, not just inventory purchases, in the numerator. This corresponds to $y_{53} + y_{58}$ in the numerator. Our approach applies to any linear combination and thus can also be used here. The approach yields 3 and 5 as bounds on $y_{53} + y_{58}$ and, hence, 0.35 to 0.59 as the redefined turnover ratio bounds. Note, in this case, the standard approximation is not even feasible when the consistency of the entire system is kept in mind.

References

Ahuja, R., T. Magnanti, and J. Orlin. 1993. *Network flows: Theory, algorithms, and applications*. Upper Saddle River, NJ: Prentice-Hall.

Arens, A., and J. Loebbecke. 2000. *Auditing: An integrated approach*. Upper Saddle River, NJ: Prentice-Hall.

Arya, A., J. Fellingham, J. Glover, D. Schroeder, and G. Strang. 2000. Inferring transactions from financial statements. *Contemporary Accounting Research* 17 (3): 365–85.

Bazaraa, M., J. Jarvis, and H. Sherali. 1990. *Linear programming and network flows*. New York: John Wiley.

Butterworth, J. 1972. The accounting system as an information function. *Journal of Accounting Research* 10 (1): 1–27.

Demski, J. 1994. *Managerial uses of accounting information*. Boston: Kluwer Academic Press.

Ijiri, Y. 1971. Fundamental queries in aggregation theory. *Journal of the American Statistical Association* 66 (336): 766–82.

Ijiri, Y. 1975. *Theory of accounting measurement*. Sarasota, FL: American Accounting Association.

Luenberger, D. 1989. *Linear and nonlinear programming*. Reading, MA: Addison-Wesley.

Mattessich, R. 1964. *Accounting and analytical methods*. Homewood, IL: Richard D. Irwin.

Pany, K., and O. Whittington. 1994. *Auditing*. Burr Ridge, IL: Richard D. Irwin.

Strang, G. 1988. *Linear algebra and its applications*. Orlando, FL: Harcourt Brace Jovanovich.

Sunder, S. 1997. *Theory of accounting and control*. Cincinnati, OH: International Thomson Publishing.

Williams, T., and C. Griffin. 1964. *The mathematical dimension of accountancy*. Cincinnati, OH: South-Western Publishing Company.