One problem in debugging a linear program is finding a way to diagnose an infeasible instance. The sources of error could be structural, such as inadvertent omission of activities, or data related, such as insufficient supply to meet demand. I present techniques that LP experts have used in practice for a variety of applications. It is important, however, to distinguish a diagnosis from an isolation. An isolation is a portion of the linear program obtained in some purposeful way to contain a probable cause. A diagnosis additionally requires an explanation of an isolation, which can require complex reasoning.

Sometimes an instance of a linear programming model results in an infeasible system of constraints. This could be due to data errors or to structural inconsistencies, and one needs to diagnose the infeasibility in order to repair the error.

Generally, one begins the diagnostic process by isolating a portion of the model believed to contain the cause of the infeasibility. Often, more than one isolation contains a plausible cause, so some reasoning is necessary to explore the alternatives. A diagnosis is an explanation of one of these isolations. In this paper, I consider three methods of isolation: (1) Phase I price aggregation, (2) irreducible infeasible subsystems, and (3) successive bound reduction. At issue is how to use the information obtained from an isolation to form a diagnosis.

In the next section, I preview these methods with simple examples. Then, I describe the three methods of isolation and apply them to infeasible instances of a
blending problem. One point to bear in mind is the distinction between isolation and diagnosis:
Diagnosis = Isolation + Explanation. This equation says that after obtaining an isolation of a portion of the linear program that contains a probable cause, we must then explain that isolation. The explanation phase can involve complex reasoning, and we can seek other isolations if the explanation is inadequate. We can use the reasons for the weakness to direct a search for a new isolation.

The quality of the information from an isolation method is not entirely quantifiable. We can, however, consider the extremes. I say the isolation yields an immediate diagnosis if little or no effort is needed to explain it. For example, suppose the LP contains an equation with only non-negative coefficients, but the right-hand side is negative and the activity levels must be non-negative. If an isolation method finds this row, the formation of a diagnosis is immediate.

I say the isolation fails if its information offers little or no clues about the cause of the infeasibility. This can happen, for example, when the optimizer terminates by detecting infeasibility with a basis whose associated activity levels contain a value that violates one of its bounds but does not reveal easily why the infeasible level cannot be driven towards the violated bound without increasing the total infeasibility. Between these extremes, I use terms like good diagnosis to mean that the information from the isolation provides a useful starting point, where we need only a modest amount of additional analysis to form a complete diagnosis that correctly identifies the cause.

At present, the meaning of correctly, or true cause, is vague, but the intent is to form a diagnosis that is meaningful to the analyst, even if the analyst is not expert in linear programming, for example, an engineer who understands the problem represented by the linear program. As I illustrate here, some of the art in forming a diagnosis can be put into the realm of science by understanding more clearly what experts do—that is, what information they seek and in what form in order to explain the infeasibility to someone, like the engineer or a data-base manager. We cannot eliminate the vagueness in all cases without understanding the particular model, but we can use some approaches that are commonly taken.

The examples I present illustrate how the cause of an infeasibility can be elusive, in that the basis at which it is detected by an optimizer need not give an immediate diagnosis. Imagine, for example, a network flow problem in which the supply is insufficient to satisfy demands. A solver can detect infeasibility and terminate with negative levels of transportation activities while the row levels are feasible. The particular transportation activities need not be links for which the supply or demand data are in error. In some cases, the cause might be inadvertent omission of activities, rather than a numerical data error.

A Preview

Before I describe the methods of isolation and illustrate their use in forming diagnoses, I consider some special cases. My purpose in doing so is to preview the analysis used for more complex cases.

Consider the transportation problem
used in Part 2 [Greenberg 1993c] of this series. Figure 1 shows the problem with the amount of supply from the second supplier revised (from 20 to 10). The total supply is now only 30, while the total demand is 35. This imbalance between total supply and demand results in an infeasible linear program. The infeasibility was detected with the solution shown, where market 1 receives only five units, short of its demand for 10 units.

The prices shown are Phase I prices, which are the rates at which the total infeasibility changes with respect to a change in the amount of a supply or a demand. For example, if the first supplier's availability is increased from 20 to 21, the total infeasibility decreases by 1. This is accomplished by shipping the additional unit to market 1.

We can form an aggregate supply constraint by summing the original supply constraints:

$$[x_{11} + x_{12} + x_{13}] + [x_{21} + x_{22} + x_{23}]$$

$$\leq 20 + 10 = 30.$$  

The first three terms in brackets are from the first supply constraint, and the next three are from the second supply constraint. Similarly, we can form an aggregate demand constraint by summing the three original demand constraints:

$$[x_{11} + x_{21}] + [x_{12} + x_{22}] + [x_{13} + x_{23}]$$

$$\geq 10 + 15 + 10 = 35.$$  

The left side of each inequality is the sum of all six flow variables, so we can observe the inconsistent constraints:

Total flow $\leq$ Total supply $= 30$, and

Total flow $\geq$ Total demand $= 35$.

The difference between the aggregate supply and the aggregate demand constraints gives a net flow constraint:

$$0 \geq \text{Total demand} - \text{Total supply}.$$  

In our case, the right-hand side is 5, so the one aggregate constraint is infeasible.

Total imbalances like that in this example are easy to check, but the situation is a bit more complicated even when such an imbalance is the basic cause of the infeasibility. A reason for complication is that network models generally have sparse

<table>
<thead>
<tr>
<th>Supply</th>
<th>Prices</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>65</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Demand:</th>
<th>Prices:</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 1: This transportation problem is infeasible because total supply is less than total demand.
Min Cut = Max Flow = 32

Figure 2: This network has no feasible flow, even though total supply equals total demand (60), because the maximum flow across the network is less than the total demand. The arcs on the left, without tail nodes, are supplies, and the arcs on the right, without head nodes, are demands. Note that total supply = 60 = total demand. The remaining arcs, which have tail and head nodes, have limited flow capacities, shown above each arc [Greenberg 1988b, Greenberg and Murphy 1991].

linkages, so that the total supply can equal (or be greater than) total demand, but the flow restrictions limit how much can be sent to each destination (Figure 2).

One isolation is a minimum cut set—that is, a set of arcs of minimum total capacity whose removal disconnects the network into two components. There is a theorem that says this is the maximum flow that can be shipped across the network from supplies to demands, so the fact that the min cut has a capacity of 32 says that we cannot satisfy all demand requirements because they total 60 units.

Although the min cut gives an isolation, it can be too large in problems of realistic size to enable a diagnosis. A better isolation is the subnetwork around node 5. Only 20 units can enter node 5 due to the capacity limits on arcs <2, 5> and <3, 5>, so the demand requirement of 30 units at node 8 cannot be satisfied. In general, we seek an isolation that localizes the infeasibility as much as possible.

Methods of Isolation

I present here three methods of obtaining information that are used to diagnose the infeasibility (for other methods, including some that address special structures, see Greenberg and Murphy [1991] and

November–December 1993 123
their references). The first is based on a slight extension of a basic theorem in linear programming [Dantzig 1963]. What this theorem says is that if the LP is infeasible, there is a one-constraint LP that is infeasible, formed by adding the rows weighted by the Phase I dual prices. I illustrated this with the transportation problem when I formed the aggregate flow constraint.

This has great appeal for diagnosis formation because, on the surface, it seems that explaining a one-constraint infeasibility is easy, at least compared to the original LP, which can have thousands of constraints.

Specifically, let \( S = \{x: L \leq x \leq U, Ax \geq b\} \) denote the constraints of the LP. To be infeasible means that \( S \) is the empty set. Let \( \pi \) be a Phase I dual solution (price vector) associated with the constraint on \( Ax \). Then, the theory says that \( \pi \geq 0 \), and there are no points in the set \( \{x: L \leq x \leq U, \pi Ax \geq \pi b\} \).

Suppose we compute the range of \( \pi Ax \), subject to the bound constraints \( L \leq x \leq U \). Theory says that the maximum in this range is less than \( \pi \beta \), so the new information obtained from the range addresses the question, Why can’t the maximum value of the aggregate constraint be at least as great as required (\( \pi \beta \))? We can compute the range of \( \pi Ax \) using the bounds on \( x \) as follows. The coefficient of \( x_i \) is \( \pi A_i \). If \( \pi A_i = 0 \), the level of activity \( j(x_i) \) does not affect the range of \( \pi Ax \). If \( \pi A_i > 0 \), the greatest value of this term is \( \pi A_i U_i \)—that is, by setting \( x_i = U_i \). If \( \pi A_i < 0 \), the greatest value of this term is \( \pi A_i L_i \)—that is, by setting \( x_i = L_i \). We reverse the levels to obtain the least value of each term in \( \pi Ax \), giving us a myopic range, \([\alpha, \beta] \) (where we could have \( \alpha = -\infty \)). The theory tells us that \( \beta \) is strictly less than the required minimum value, \( \pi b \).

In general, many of the aggregate coefficients can be zero, and slack constraints have zero prices. In fact, it is possible for all aggregate coefficients to be zero, in which case the aggregate constraint is of the form: \( 0 \geq \pi b \), while \( \pi b > 0 \). In this case, we have no new information. Only when we obtain some nonzero coefficients can the aggregate be useful in diagnosis formation.

Consider the infeasible network flow problem in Figure 2. Algebraically, the constraints are as follows, where \( x_{ij} \) is the flow from node \( i \) to node \( j \):

Supply at node 1: \( x_{14} \leq 20 \),
Supply at node 2: \( x_{24} + x_{25} \leq 20 \),
Supply at node 3: \( x_{35} \leq 20 \),
Balance at node 4:
\[ x_{14} + x_{24} - x_{46} - x_{47} = 0, \]
Balance at node 5:
\[ x_{25} + x_{35} - x_{57} - x_{58} = 0, \]
Demand at node 6: \( x_{46} \geq 10 \),
Demand at node 7: \( x_{47} + x_{57} \geq 20 \),
Demand at node 8: \( x_{58} \geq 30 \), and
Flow Bounds: \( 0 \leq x_{14} \leq 30 \), \( 0 \leq x_{24} \leq 20 \),
\[ 0 \leq x_{25} \leq 10, 0 \leq x_{35} \leq 10, \]
\[ 0 \leq x_{46} \leq 10, 0 \leq x_{47} \leq 2, \]
\[ 0 \leq x_{57} \leq 20, 0 \leq x_{58} \leq 30. \]

MINOS detected infeasibility at a basis having the flows shown in Figure 3. Notice
that only one row constraint is infeasible: supplier 1 ships 30 units, which is 10 more than the availability. There are, however, two flow infeasibilities: \(x_{25}\) and \(x_{47}\) exceed their capacities.

To form an infeasible aggregate constraint, we multiply each row constraint by its Phase I dual price and sum. Figure 4 shows the prices (plus other information). Before forming the aggregate, consider what these prices mean.

Row NODE1 represents the first supply constraint, and it is at the infeasible level of 30. The dual price of \(-1\) says that if the supply limit is increased from its current value of 20, the total infeasibility will decrease at a unit rate. Suppose, for example, we change the supply limit to 21. Then, this supplier’s level is only nine units above its limit, rather than its current excess of 10.

Row NODE2 also has a dual price of \(-1\) even though its level (20) is feasible. What would happen if we increase its supply limit to 21 is that flow from node 2 to node 4 can be increased to one unit. This would displace flow from node 1 to node 4, thereby decreasing the infeasible level of the first supplier.

The zero dual price for row NODE3 reflects the fact that this supplier has a surplus. Making more supply available at
Figure 4: This is what an analyst sees when he or she displays the rows and columns in the infeasible network of Figure 2.

Node 3 has no effect on the infeasibility.

The next two rows are the balance equations at nodes 4 and 5. Unlike the supply constraints, this is a logical relation that follows from the network’s topology, not from data. We cannot change the requirement that the inflow minus the outflow must equal zero. Still, the prices tell us that we can decrease the total infeasibility by making the right-hand side negative. This would correspond to providing stockpiles at nodes 4 and 5, which is then a new source of supply.

Row NODE6 is the first demand requirement. Its dual price of 1 says that we can decrease the total infeasibility by decreasing the demand requirement. If, for example, we change the demand to be only nine units, we can decrease the flows along the delivery path, 1 → 4 → 6, by one unit. This decreases the level of flow out of node 1, which thus decreases total infeasibility by one unit.

Rows NODE7 and NODE8 have dual prices of 2 because their delivery paths have two infeasibilities that are decreased. For NODE7 the delivery path is 1 → 4 → 7. In addition to decreasing the flow out of supplier 1, the flow across arc <4, 7> is decreased by one unit as well. The current level is $x_{47} = 20$, which is infeasible because it exceeds its capacity of 2. The de-
livery path for node 8 is 2 → 5 → 8, so a decrease in the last demand decreases the flow across arc (2, 5), which is currently infeasible. An additional decrease in infeasibility comes from decreasing the flow out of supplier 2. We have already seen that if another unit is available from supplier 2, this decreases the infeasible level of supplier 1, so the net effect is to decrease total infeasibility at a rate of two units per unit decrease in the demand requirement at node 8.

Tracing flows, as we did in this example, is an exercise in price interpretation, which we considered in Part 2 of this series. Our task is to diagnose the infeasibility, and price interpretation is the basis for one approach to isolate a portion of the linear program. In practice, however, there will be many nonzero dual prices, and it is not clear which one to pick to begin a flow trace. The theory tells us to look at the aggregate formed by multiplying each row constraint by its dual price. For this example, the result is the following aggregate constraint.

\[ x_{25} + 2x_{35} + x_{47} \geq 70. \]

The aggregate row has only three nonzero coefficients, and these particular flow variables provide an isolation we seek to explain. The myopic range in this case is [0, 32], which must be disjoint from the required range, [70, \(\infty\)]. The myopic range is determined as follows. First, since each flow variable is required to be non-negative and the coefficients are positive, its least value is zero. The greatest value of 32 comes from the capacity constraints, which are upper bounds on the levels of flows. The first term, \(x_{25}\), is bounded by 10 units; the second term, \(2x_{35}\), is bounded by 20 units (from the capacity constraint, \(x_{35} \leq 10\)); and, the third term, \(x_{47}\), is bounded by two units. The total value, therefore, is bounded by 32.

In summary, this isolation to the three flow variables can be used to formulate the following diagnosis. The constraints imply \(x_{25} + 2x_{35} + x_{47} \geq 70\), but the flow capacities limit the left-hand side to only 32 units — that is, \(x_{25} \leq 10, x_{35} \leq 10, \) and \(x_{47} \leq 2\) imply \(x_{25} + 2x_{35} + x_{47} \leq 32\).

Despite the appeal of this simple approach, the examples will show that a useful explanation might not be immediate. Besides the possibility that all aggregate coefficients might be zero (giving us no new information whatsoever), we could also find that, for the explanation to be “useful,” we must explain the prices in terms of original data. The information presented above does not say why this constraint is implied by the original system of constraints. In particular, why are these the only flow variables in the aggregate, and why is 2 the coefficient of \(x_{35}\)?

The primary information that is potentially useful in forming a diagnosis is which activities have nonzero coefficients in the aggregate row, not necessarily the numerical values of these coefficients. This method yields an immediate diagnosis, as in the example, when the number of nonzero aggregate coefficients is at least one and there are not too many to comprehend.

The second method of diagnosis isolates a portion of the model by directly seeking an infeasible subsystem that is irreducible—that is, no proper subsystem is infeasible. All bound constraints, including sim-
ple non-negativity, are included in the formation of this set. This method was originally suggested by Debrosse and Westerberg [1973]; van Loon [1981] went further into its development for linear programs. Chinneck [1990, 1991] gives the most extensive algorithmic development of this approach (see, also, Chinneck and Dravnieks [1991], and see Greenberg and Murphy [1991] for a basic description in context with other methods).

Let us reduce the infeasible network in Figure 2 to an infeasible subsystem. We can do this by discarding constraints having a zero price. Using the Phase I dual prices, which are displayed in Figure 4, we obtain the following infeasible subsystem:

Supply at node 1: \( x_{14} \leq 20 \),

Supply at node 2: \( x_{24} + x_{25} \leq 20 \),

Balance at node 4:

\[ x_{14} + x_{24} - x_{46} - x_{47} = 0, \]

Balance at node 5:

\[ x_{25} + x_{35} - x_{57} - x_{58} = 0, \]

Demand at node 6: \( x_{46} \geq 10 \),

Demand at node 7: \( x_{47} + x_{57} \geq 10 \),

Demand at node 8: \( x_{58} \geq 30 \),

Flow bounds: \( x_{25} \leq 10, x_{35} \leq 10, x_{47} \leq 2 \).

This subsystem has all row constraints (supply, balance, demand), except the supply limit at node 3. Only three of the capacity constraints are in the subsystem, and it is still infeasible, even without non-negativity constraints. Graphically, the network looks like the original, except there is unlimited supply into node 3 and most of the capacity values are absent (indicating no bound).

Although this is a reduction of the original infeasible system of constraints, it can be reduced further, and we can obtain a much smaller subnetwork. In particular, the following subsystem of constraints is infeasible.

Balance at node 5:

\[ x_{25} + x_{35} - x_{57} - x_{58} = 0, \]

Demand at node 8: \( x_{58} \geq 30 \),

Flow Bounds: \( x_{25} \leq 10, x_{35} \leq 10 \).

This reduction is significant, and look how small the graphical display becomes.

The two arcs into node 5 correspond to the original arcs \( \langle 2, 5 \rangle \) and \( \langle 3, 5 \rangle \), but now there is infinite supply, dropping nodes 2 and 3 from the network. The capacities of these arcs are limiting the flow into node 5 to be no more than 20 units. The arc \( \langle 5, 8 \rangle \) has no capacity limit in this subnetwork, having dropped the capacity constraint in forming this infeasible subsystem. The balance equation (row NODE5) remains in the system, as does the demand requirement out of node 8.

Clearly the reduction paid off in this example. The isolation is very localized around node 5, enabling a diagnosis that is better than the first reduction associated with nonzero dual prices. It is the quality of being irreducible that makes this infeasible subsystem a better localization than the
previous one, obtained from nonzero Phase I prices.

Chinneck and Dravnieks [1991] have developed a variety of algorithms to obtain an irreducible infeasible subsystem (IIS), which differ by speed of computation. One greedy algorithm, which was the one used for this study, is to drop constraints until the subsystem becomes feasible. Then, the last one dropped is added back. Keeping this constraint in the subsystem, the others are interrogated to see if they can be dropped (one at a time). After one pass through the others, the resulting subsystem is irredicibly infeasible.

A diagnosis can be formed from an IIS merely by displaying the constraints and explaining that they cannot hold simultaneously. Depending upon the LP knowledge of the analyst, this could be sufficient, or it could require more explanation. In any case, some constraint (that is, equation or bound) in the IIS must change for the problem to become feasible if the body of the LP is presumed structurally correct (that is, the sign pattern is correct). Without this presumption, such as the case of inadvertent omission of activities, the IIS can still provide information that is a good starting point to obtain a diagnosis. The experiments, given in the subsequent sections, use Chinneck’s [1991] code to generate an IIS for infeasible instances of a blending model. Then, in each case, I use ANALYZE [Greenberg 1987a, 1987b, 1988a, 1992a, 1993a, 1993d] to read this and provide diagnosis formation.

The third method I consider is that of successive bounding, which has an early history in preprocessing linear programs. Simple tests are applied iteratively to infer tighter bounds on the activities and on the range constraints. In some cases, the method detects infeasibility, perhaps after many iterations with other reductions that are irrelevant to its cause.

For example, consider the following $2 \times 3$ system.

$$S = P - T = 0, \quad D = T - C = 0,$$

$$0 \leq P \leq U, \quad 0 \leq T, \quad L \leq C.$$ 

Think of the activities as production ($P$), transportation ($T$) and consumption ($C$); and, think of the equations as supply ($S$) and demand ($D$) balances. Then, $U$ is a bound on production and $L$ is a requirement on consumption. This is infeasible if $L > U$.

By the method of successive bounding we would infer, from the supply equation and the production bound, that $T \leq U$ (by simply recognizing $T = P \leq U$). Then, when the method computes the myopic range of the demand equation, it finds that its greatest value is $U - L$ (with $T = U$, its inferred upper bound, and $C = L$, its original lower bound). At this point the method detects infeasibility since $U - L < 0$, while the equation requires 0. This could be part of a larger LP, so this subsystem is an isolation. Unlike the other isolation methods, the successive bounding gives a causal chain. In this case, we would see something like the following for $U = 10$ and $L = 100$.

1. Equation $S$ and $P <= 10$

   imply $T <= 10$.

2. $T <= 10$ and $C >= 100$

   imply $D <= -90$.
The first implication uses equation 5 and the bound on $p$ to infer the same bound on $T$. With this inferred bound, $T \leq 10$, plus the lower bound on $C$, the method determines that the maximum value of $D$ cannot exceed $-90$, which is infeasible since we require $D = 0$.

In general, if the method of successive bound reduction detects infeasibility, it is with a row’s range constraint, perhaps after some of the other bounds have been tightened. We obtain a diagnosis by finding a causal chain of bound tightening that led to the detection.

As I shall illustrate with examples, the causal chain we infer from the sequence of reductions leading to the detection of infeasibility is more than an isolation. It is itself a diagnosis. The approach, however, can fail to detect infeasibility. This depends on the particular tests performed and on the source of the infeasibility.

**Examples of Diagnosis Formation**

To consider infeasible instances of a blending problem, I use the simplest class of blending models that were used by Greenberg [1992b] (which also considered regional and dynamic blending models).

Figure 5 illustrates the model structure. There are two types of processing units: primary and blending. A primary unit takes as its input raw feed stock, and it produces blend stocks and final products. A blending unit takes, as its input, blend stocks, and it produces final products. Both unit types have capacity limits and costs of operation.

The algebraic description of this blending problem is as follows:

Minimize \( \text{COST} = \sum_{pr} CP_{pr} OP_{pr} + \sum_b CB_b OB_b \),

subject to \( OP, OB \geq 0 \),

Capacity limits = \( OB_b \leq \text{CAP}_{-B_b} \);

\( \sum_r OP_{pr} \leq \text{CAP}_{-P_r} \),

Raw feed stock = \( \sum_{r} FR_{pr} OP_{pr} \leq \text{SUPPLY}_{r} \),

Blend stock balance = \( \sum_r BS_{pr} OP_{pr} - \sum_b FB_{sb} OB_b = 0 \),

Demand = \( \sum_{pr} PP_{pr} OP_{pr} + \sum_b PB_{sb} OB_b \geq \text{DEMAND}_{f} \),

where the index domains are

- \( p \) is in a set of primary units (PLU),
- \( b \) is in a set of blend units (BU),
- \( r \) is in a set of raw feed stocks (RF),
- \( s \) is in a set of blend stocks (BS),
- \( f \) is in a set of final products (FP),

the data tables are

---

**Figure 5:** This simple blending model is used to illustrate some infeasible structures.
LINEAR PROGRAMS

CP = cost of operating primary unit,  
CB = cost of operating blend unit,  
CAP_P = capacity of each primary unit,  
CAP_B = capacity of each blend unit,  
FR = raw feed stocks into each primary unit,  
FB = blend stocks into each blend unit,  
BS = blend stocks from each primary unit,  
PP = final products from each primary unit,  
PQ = final products from each blend unit,  
SUPPLY = supplies of raw feed stocks,  
DEMAND = demands for final products,  
the decision variables are  
OP = operation levels of primary units,  
OB = operation levels of blend units.

Figure 6 gives an overview of an infeasible instance of the blending model. The schema shows not only the syntax of the linear program but also the coefficient ranges in each cell. For example, the range of coefficients in the submatrix composed of balance rows (BAL) and activities that operate primary units (OP) is the interval, [0.1, 0.6]. In reference to the algebraic model description, this is the range in table BS. The range of nonzero coefficients in the submatrix composed of raw material supply rows (SUP) and OP columns is 1, which reflects the model’s logic—that is, it is not dependent upon data values. The bound values are included (last two rows, labeled :LO and :UP), and the asterisk (*) denotes infinity.

The reason we begin with the schema display is that sometimes it can reveal an error in some coefficient range; or that some nonzeros were inadvertently omitted entirely, which appears as a blank cell (as in the COST row, but this is irrelevant to the infeasibility). In this case, nothing appears unusual, so we proceed to apply the three methods of isolation that I described in the previous section.

Names of specific rows and columns are formed from the syntax in Figure 6 (see Part 1 of this series [Greenberg 1993b]). For example, if we have a primary unit 02 (member of PU) and a raw feed HC (member of RF), the name of the primary operation activity (in class OP) is OP02HC. Similarly, OP04HC is the name of the activity that operates primary unit 04 using raw feed HC.

| ::::: ::::: BLOCK SCHEMA ::::: |
| OP(PO,RF)  | OB(BU)   | RHSMODL |
| SUP(RF) 1   | <= 20    |
| CAP(PO) 1   | <= 10    |
| BAL(BS) 0.1/0.6 | -1/-0.1  | = 0     |
| DEM(FP) 0.1/0.5 | 0.1/1    | >= 5/15 |
| COST        | 0/0      |
| :LO 0       | 10/15    |
| :UP *       |          |

Figure 6: This is one overview of an infeasible instance of a blending problem that an analyst can see with the ANALYZE system.
Figure 7 shows the aggregate constraint using Phase I prices as weights. Of the four activities with nonzero aggregate coefficient, only two are relevant to the diagnosis formation. The non-negativity of the primary unit operation activities, namely OP02HC and OP04HC, are logical lower bounds, rather than data dependent. What this display tells us is that the upper bounds on the two blend unit operation activities, OB01 and OB04, are the probable cause. This is because the information reveals not only that the aggregate required range is disjoint from its myopic range (which the theory already says must be true), but also that the reason the myopic range cannot exceed 15.966 is because the upper bounds on these two activities sum to this total.

To elaborate, we want to form a diagnosis from this information. We see that the max of the myopic range (15.966) is less than the min required (32.373), so we use the right-most column to address the question, Why can’t the aggregate requirement of 32.37 be achieved? The upper bounds of zero for the last two activities come from the non-negativity of their levels plus the fact that their aggregate coefficients are negative. We can reduce the isolation by regarding non-negativity as a logical constraint, not dependent on the data, and write the implied aggregate constraint:

\[0.8055 \text{OB01} + 0.7911 \text{OB04} \geq 32.37.\]

The left side of this constraint has a maximum value (upper limit of its myopic range) of 15.966 because each of the two activities have upper bounds of 10. Thus, the capacity to operate blend units 01 and 04 limits the aggregate generation to only 15.966, which is less than the required amount (32.373).

This diagnosis might suffice, but one might also ask for an explanation of these aggregate coefficients. In that case, this information is only a starting point for further analysis. In general, the information explains the myopic range; it answers the question, Why can’t the value of the aggregate row be greater? One could be more concerned with why the aggregate row is required to be at least 32.373 and why this row is implied by the constraints. Despite this caveat, the diagnosis formed in this example represents a reasonable diagnosis of the infeasibility.

Let us review the reasoning in using the

<table>
<thead>
<tr>
<th>Aggregate ROW range: 32.373 to *</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficient:</td>
<td></td>
</tr>
<tr>
<td>0.8055381 OB01 0</td>
<td>8.055</td>
</tr>
<tr>
<td>0.7910840 OB04 0</td>
<td>7.911</td>
</tr>
<tr>
<td>-0.3388400 OP02HC -*</td>
<td>0</td>
</tr>
<tr>
<td>-0.9478710 OP04HC -*</td>
<td>0</td>
</tr>
<tr>
<td>...Myopic range: -*</td>
<td>to 15.966</td>
</tr>
</tbody>
</table>

Figure 7: This is what an analyst sees with ANALYZE computing the Phase I aggregate infeasible constraint for an instance of the blending problem.
information obtained to form a diagnosis. First, the reasoning involved getting the nonzero coefficients of the aggregate constraint. Second, with the maximum of the myopic range less than the minimum requirement, we formed the set of activities whose bounds cause the limit in the maximum value that can be achieved. Third, upper values of zero that come from non-negativity constraints (and negative aggregate coefficients) were dropped, leaving the capacities of the remaining activities as the cause of the infeasibility. These steps isolated two activities (OB01 and OB04), and the diagnosis gave their capacities as the cause.

Now consider the IIS method of isolation. Figure 8 shows the IIS rows, and the IIS column bound constraints are the same ones we obtained from the aggregation:

\[
\begin{align*}
0 &= \text{BALGO} = -0.3 \text{ OB02} - 0.7 \text{ OB03} - 0.3 \text{ OB04} \\
  &\quad + 0.3 \text{ OP01MC} + 0.5 \text{ OP02MC} + 0.4 \text{ OP03MC} \\
0 &= \text{BALHN} = -0.7 \text{ OB01} - 0.7 \text{ OB02} - 0.2 \text{ OB04} \\
  &\quad + 0.4 \text{ OP04MC} + 0.1 \text{ OP05MC} \\
0 &= \text{BALLN} = -0.3 \text{ OB01} - 0.1 \text{ OB04} + 0.1 \text{ OP04MC} \\
  &\quad + 0.5 \text{ OP05MC} \\
0 &= \text{BALRS} = -\text{ OB05} - 0.3 \text{ OB03} - 0.4 \text{ OB04} \\
  &\quad + 0.6 \text{ OP01MC} + 0.35 \text{ OP02MC} + 0.15 \text{ OP03MC} \\
10 &\geq \text{CAP02} = + \text{ OP02HC} + \text{ OP02MC} \\
10 &\geq \text{CAP03} = + \text{ OP03HC} + \text{ OP03MC} \\
10 &\geq \text{CAP04} = + \text{ OP04HC} + \text{ OP04MC} \\
10 &\geq \text{CAP05} = + \text{ OP05HC} + \text{ OP05MC} \\
10 &\leq \text{DEMFO} = + \text{ OB05} + 0.2 \text{ OB01} + 0.1 \text{ OB02} + 0.2 \text{ OB04} \\
  &\quad + 0.1 \text{ OP03HC} + 0.1 \text{ OP03MC} + 0.1 \text{ OP04HC} \\
  &\quad + 0.1 \text{ OP04MC} + 0.2 \text{ OP05HC} + 0.2 \text{ OP05MC} \\
15 &\leq \text{DEMRE} = + \text{ OB01} + 0.4 \text{ OB03} \\
  &\quad + 0.5 \text{ OB04} \\
20 &\geq \text{SUPMC} = + \text{ OP01MC} + \text{ OP02MC} + \text{ OP03MC}
\end{align*}
\]

Discarding the non-negativity, as we did in the aggregation method, we can arrive at the same diagnosis. First, we examine the balance rows to make sure they are formed properly. Satisfied that this is the case, we recognize the balance equations limit operation levels, but our attention is now drawn to the capacity limits, the demand requirements and the one supply limit in the IIS. We could perform the arithmetic that shows the demands cannot be satisfied with the capacities and supply values in this instance.

One must think about whether to let this be the diagnosis or probe deeper into why these are inconsistent. If the diagnosis given is acceptable, the time to form it is

Figure 8: An analyst can list the IIS equations with ANALYZE.
slightly more with the IIS approach due to having too much information (from the view of automating the reasoning, information overload can hinder heuristics designed to automate the diagnosis formation). If, on the other hand, a deeper analysis is required, the greater information from the IIS approach gives a better starting point to begin the investigation.

It helps to relate the Phase I aggregation approach with the IIS approach. An infeasible subsystem can be formed from the Phase I prices by including all constraints with nonzero Phase I price. This includes not only the rows, but also bounds on the columns, whose associated reduced cost is the dual price for its bound constraint.

Such an infeasible subsystem can be reducible—that is, it can contain a subset of constraints that is also infeasible. The IIS, by its definition, is an infeasible subsystem that cannot be reduced. Thus, an IIS can be formed by seeking reductions from this infeasible subsystem formed by Phase I prices (which is one of Chinneck’s methods). In fact, the Phase I isolation can be a union of many IISs.

The third method of isolation is successive bounding (besides ANALYZE, most commercial optimizers have a preprocessor that performs successive bounding). If infeasibility is detected, it is with a range constraint for an equation, usually after some of the other bounds have been tightened. A diagnosis is obtained by finding the causal chain of bound tightening that led to the detection.

Figure 9 shows that infeasibility detection occurred with row BALHN (which balances heavy naphtha). The bound tightening resulted in a computed (myopic) value range of $[-11.133, -.2095]$. This means this row cannot reach its required level of zero.

The question is, given what you see in Figure 9, how do we proceed to form a diagnosis? We begin the analysis by looking row at BALHN, which is listed in Figure 10. The original bounds on the five activities that are in the equation do not explain the infeasibility—that is, this equation could be satisfied with the original bounds: $0 \leq OB01$, $OB04 \leq 10$, $0 \leq OB02 \leq 15$, and $0 \leq OP04MC$, $OP05MC$. What we

\[
0 = BALHN = -0.7 \ OB01 - 0.7 \ OB02 - 0.2 \ OB04 + 0.4 \ OP04MC + 0.1 \ OP05MC
\]

Figure 10: The analyst lists the equation (BALHN) that ANALYZE detected to be infeasible.
Blend stock heavy naphtha cannot be balanced due to the following.

1. \( OB01 \leq 10, OB03 \leq 12, OB04 \leq 10 \) and Row DEMPG imply \( OB02 \geq 0.6667 \).
2. Row CAP04 implies \( OP04MC \leq 10 \).
3. \( OP04MC \leq 10, OB01 \leq 10, OB04 \leq 10 \), and Row BALLN imply \( OP05MC \leq 8 \).
4. \( OP04MC \leq 10, OP05MC \leq 8, OB02 \geq 0.6667 \), and Row BALHN imply \( OB01 \leq 6.19 \).
5. \( OP04MC \leq 10, OP05MC \leq 8, OB02 \geq 0.6667 \), and Row BALHN imply \( OB02 \leq 6.857 \).
6. \( OB02 \leq 6.857, OB03 \leq 10, OB04 \leq 10 \), and Row DEMRG imply \( OB01 \geq 1.81 \).
7. \( OB01 \leq 6.19, OB03 \leq 10, OB04 \leq 10 \), and Row DEMPG imply \( OB02 \geq 3.206 \).
8. \( OB02 \leq 6.857, OB03 \leq 10, OB04 \leq 10 \), and Row DEMPG imply \( OB04 \geq 6.349 \).
9. \( OB01 \geq 1.81, OB04 \geq 6.349, OP04MC \leq 10 \), and Row BALLN imply \( OP05MC \leq 5.714 \).

Now the max value of row BALHN is determined with these bounds as:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Coefficient</th>
<th>Relevant Bound</th>
<th>Max Value of Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>OB01</td>
<td>-0.7</td>
<td>( L = 1.81 )</td>
<td>-1.267</td>
</tr>
<tr>
<td>OB02</td>
<td>-0.7</td>
<td>( L = 3.206 )</td>
<td>-2.244</td>
</tr>
<tr>
<td>OB04</td>
<td>-0.2</td>
<td>( L = 6.349 )</td>
<td>-1.27</td>
</tr>
<tr>
<td>OP04MC</td>
<td>0.4</td>
<td>( U = 10 )</td>
<td>4</td>
</tr>
<tr>
<td>OP05MC</td>
<td>0.1</td>
<td>( U = 5.714 )</td>
<td>0.5714</td>
</tr>
</tbody>
</table>

\[ \text{Sum} = -0.2096. \]

Figure 11: The analyst sees the causal chain and relevant bounds, tightened by the ANALYZE successive bound reduction method. This provides a diagnosis of the infeasibility.

must investigate is why some of the bounds are tightened, which we have previously called a causal chain.

The causal chain (implications 1-9 in Figure 11) is obtained from the REDUCE log. The general structure of the diagnosis in Figure 11 begins with a translation of the row where infeasibility was detected by REDUCE. This is what appears as the first sentence in the above diagnosis, where the row is BALHN. Then, the causal chain is given. The last part of the diagnosis is a summary table of the final bounds and how they contribute to the computed range of row BALHN. The coefficients are the original ones, which is an important distinction with the information provided by Phase I aggregation.

The first inference of the causal chain refers to row DEMPG, shown in Figure 12. This demand requirement gives a bound on activity OB02:

\[ 0.3 \ OB02 \geq 10 - 0.2 \ OB01 - 0.4 \ OB03 - 0.3 \ OB04. \]

Then, the upper bounds on the activities (as stated in Figure 12) imply that the right-hand side is at least \( 10 - 0.2(10) - 0.4(12) - 0.3(10) \), which is 0.2. This gives the first implication of the causal chain: \( OB02 \geq 0.2/0.3 = 0.6667 \).

Each of the subsequent implications uses the same reasoning. Variable bounds, which may be original or previously inferred, plus an equation yield a tighter
bound on some variable. The chain of implications leads to the summary table at the bottom, showing why the max of row BALHN is only $-0.2096$, thus diagnosing the infeasibility.

Figure 13 gives a schematic of the first level of causality composed of the final, relevant bounds that were inferred. The implication is shown as an input to the inferred bound, which is boxed. The bound, OP04MC $\leq 10$, is an original bound, so there is no implication statement in that box.

We can walk through the causal chain by expanding some implication. For example, if we want to know why OB01 $\geq 1.81$, we can draw a schematic of implication 6:

![Diagram of implications]

The entire causal chain is an implication network, which has a natural hierarchy. It may, however, be too large to comprehend at once, so a walk through each level is useful. This allows us to focus on one im-

Figure 13: A schematic of the bottom level of reasoning for the causal chain in Figure 11 can help an analyst walk through the inferred bounds graphically.
Figure 14: This is what an analyst could see when ANALYZE computes the Phase I aggregation. The fact that all aggregate coefficients are zero is a failure of the Phase I aggregation method in this case.

plication level (or sequence) at a time, starting with the inferred bounds that gave the infeasibility detection for some row.

Each method of isolation can fail in the sense that it need not provide information appropriate to enable a diagnosis. To understand why this is so, bear in mind the reasoning we have used to form a diagnosis from the information obtained by each isolation method.

Successive bounding can fail by not detecting infeasibility at all. When it does detect infeasibility, and when the causal chain is not confounded by a large number of irrelevant reductions, its information enables a deeper diagnosis than both Phase I aggregation and an IIS.

Phase I aggregation fails to provide new information when all aggregate coefficients are zero. What do you do when you face a screen like that shown in Figure 14? We see the aggregate is required to be at least 1, and its maximum achievable value is 0, but there are no activities in the aggregate row; we just have the constraint \( 0 \geq 1 \), with no clue about why the cancellations took place.

In this case the infeasibility was formed by augmenting a linear combination of supply rows with an inconsistent right-hand side. The Phase I aggregation method must fail for this kind of infeasibility because all aggregate coefficients must be zero when a row is a linear combination of other rows.

The rule that uses the information from Phase I aggregation does not fire due to having no nonzero coefficients to explain. (Successive bounding failed to detect infeasibility in this case, so it too failed.)

Now consider the IIS isolation, shown in Figure 15. Mathematically, the IIS just obtains the isolation; it does not go further in finding out that row AGGSUP is a linear combination of the other rows in the submatrix. We can see that AGGSUP = -(SUPHC + SUPMC), so -41 \( \geq \) AGGSUP is inconsistent with SUPHC + SUPMC \( \leq 40 \).

We can interpret row AGGSUP as a re-

\[
\begin{align*}
-41 &\geq AGGSUP = OP01MC - OP02MC - OP03MC - OP04HC - OP05HC \\
20 &\geq SUPHC = OP04HC + OP05HC \\
20 &\geq SUPMC = OP01MC + OP02MC + OP03MC
\end{align*}
\]

Figure 15: The analyst begins to form a diagnosis by first listing the IIS rows for the infeasible instance of an aggregate operation requirement (row AGGSUP).
qurement that the five operation activities must have an aggregate level of operation of at least 41. This is inconsistent with the crude oil supplies that limit their total operation levels to 40 (rows SUPHC and SUPMC). Finding this relation is a matter of arithmetic, but knowing to look for it is a matter of reasoning.

The IIS gave us a good starting point by isolating these three constraints. The final diagnosis, however, requires an explanation of why they are inconsistent with each other. The clue that a linear combination exists to provide an explanation comes from the generation of all zero coefficients in the aggregate. Thus, these two methods can combine to contain sufficient information to form a diagnosis.

Summary and Conclusions

There is no universal method of debugging an infeasible linear program. Although the three methods presented here give different information, the Phase I aggregation and IIS methods of isolation are based on variants of Farkas' lemma. Successive bounding is very different and gives a causal chain if it succeeds. We can view these three methods of isolation as engines, which can be called by a rule base. This rule base can decide the appropriate sequence of calls to the engines and control the diagnosis formation with an explanation of the information obtained. As a model matures, experience can influence the control strategy through learning mechanisms. What is important is that all engines must be available; no one engine can necessarily enable a useful diagnosis in all cases.

In conclusion, the ANALYZE system comes with a variety of infeasible linear programs, some of which are exercises in the primer. It also contains an extended library that includes the blending cases I have considered here.

Acknowledgments

I gratefully acknowledge encouragement and technical help from Frederic H. Murphy, Ketron Management Science and Michael Saunders provided MINOS for solving the linear programs. John Chinneck provided MINOS(IIS) to compute irreducible infeasible subsystems. In addition, support for the ongoing project that produced ANALYZE (among other things) comes from a consortium of companies: Amoco Oil Company, IBM, Shell Development Company, Chesapeake Decision Sciences, GAMS Development Corp., Ketron Management Science, MathPro, Inc., and Maximal Software, Inc.

References

Greenberg, H. J. 1987a, "Computer-assisted analysis for diagnosing infeasible or unbounded linear programs," Mathematical Pro-
gramming Studies, Vol. 31, pp. 79–97.