Rule-based intelligence to support linear programming analysis *

Harvey J. Greenberg

University of Colorado at Denver, Denver, CO, USA

Increased size and complexity of linear programs make it difficult to understand results and manage the model. This difficulty is overcome, at least partially, with new advances in computing environments and techniques from artificial intelligence and operations research. A software system, called ANALYZE, is presented within the larger context of developing an artificially intelligent environment for mathematical programming modeling and analysis. With heuristics governed by rule-based reasoning and with syntax-driven translations into English, the ANALYZE system enables a form of intelligence to support analysis. Particular problems that illustrate this capability are explanations of dual prices, diagnoses of infeasibilities, and reasoning about redundancy.

Keywords: Linear programming, Computer-assisted analysis, Artificial intelligence, Large-scale modeling, Infeasibility, Redundancy, Duality

Introduction and background

For 40 years mathematical programming has been a powerful technique to support operational and planning decisions. Because of its wide scope of applications, linear programming (LP) is used by major industries to support their decision-making. It became clear by the late 1950’s that optimizers had to expand into database management and related support for modeling and analysis. This is when the Mathematical Programming System (MPS) took shape.

Due to the explosive growth of inexpensive computer power and to the highly successful applications during the 1960’s, we can solve far larger problems that we can understand. This is what prompted the initiation of a project to develop an Intelligent Mathematical Programming System (IMPS), which is sponsored by a consortium of industries. It is aimed at reconciling this information bottleneck and markedly extending

Harvey J. Greenberg received his Ph.D. in Operations Research from The Johns Hopkins University in 1968. He has been on the faculty of Computer Science at Southern Methodist University and at Virginia Polytechnic Institute and he was Division Director at the U.S. Department of Energy. Currently, Dr. Greenberg is Professor of Mathematics at the University of Colorado at Denver. With more than 70 research publications and 8 books (5 edited), Dr. Greenberg has pioneered computer-assisted analysis, for which he received the first ORSA/CSTS Prize for excellence in research in the interface between operations research and computer science. This has evolved into an Intelligent Mathematical Programming System, which uses various reasoning mechanisms and natural language discourse to assist building and analyzing mathematical programming models for decision support. Dr. Greenberg has given many professional services to ORSA, ACM/SIGMAP and Mathematical Programming Society/Committee on Algorithms. He has served on the editorial boards of Operations Research, Communications of ACM, Annals of Mathematics of Artificial Intelligence, Operations Research Letters, American Journal of Mathematical and Management Sciences, and has guest edited issues of Annals of Operations Research. He is the founding editor of ORSA Journal on Computing.

* This research was supported by a consortium of companies: Amoco Oil Company, General Research Corporation, Shell Development Company, Chesapeake Decision Sciences, Inc., GAMS Development Corp., Ketron Management Science, MathPro, Inc., and Optimum Methods, Inc.

Correspondence to: H.J. Greenberg, Department of Mathematics, Campus Box 170, University of Colorado at Denver, P.O. Box 173364, Denver CO 80217-3364, USA.

0167-9236/92/$05.00 © 1992 – Elsevier Science Publishers B.V. All rights reserved
computer assistance for model formulation, management and analysis. Some of the issues have been described by Greenberg [8,9,10]; Bonczek, Holsapple and Whinston [1]; Nance and Balci [26]; and, Mitra [23].

The main building blocks for the development of an IMPS are formulation, analysis and discourse. The concepts on which they are built, the functions they serve, and the techniques employed comprise the cornerstone of this foundation. These elements are shown in fig. 1 from a software vantage. The modularity continues beneath the high-level components shown.

Analysis and formulation are two intertwined modules of an IMPS, and both require discourse models with varied functionality. Discourse is how a human being communicates with the computer, and our objective is to orient this towards the human being's problem domain rather than the traditional computer language. The IMPS project has developed a foundation based on a triad of text, graphics and algebraic forms of discourse. The discourse is not only between the human expert and the IMPS, but also between the model's constituency and the decision support system generated by the IMPS. Under one framework, there are levels of discourse with different functions. One function is to communicate a problem and have a formulation result; another function is to communicate the meaning of model elements and results of instances. The latter extends to support analysis that we illustrate in this paper.

The formulation component is implemented as two primary modules: MODLER [15] and LP-FORM [25]. Both subdivide and have been implemented differently. The discourse component is in the middle because it is used by all components directly and indirectly. Beneath the primary modules there is a learning component, which is further subdivided according to the type of learning appropriate to the object of the learning. Discourse, for example, can include a vocabulary that can expand. Formulation is based on associations for which neural networks apply [13]. Analysis is rule-based. These learning mechanisms, although functionally divided, are tied through the knowledge base design and additional inference mechanisms [24,14].

Here we focus on the analysis component, implemented in a module called ANALYZE [11,16]. This provides analysis support at different levels, including a rule-base whose foundation is built upon modern insights into methods of sensitivity analysis, both quantitative and qualitative. Its English translation support is based on model syntax [12] – that is, rules of composition over domains, which has been in practice since the beginnings of linear programming.

The rest of this paper is divided into five sections. In the next section we present some technical background, including the syntax of a linear program, which is used for English translations. Then, we illustrate analysis support for three problems:
- Explain a dual price.
- Diagnose an infeasibility.
- Explain a redundancy.

In each case we go through details of analysis as a linear programming expert, using ANALYZE commands to get information. Our purpose is not only to illustrate the software, but also, more importantly, to go through the reasoning process that is then automated. The structure of a rule file in ANALYZE uses the same ANALYZE commands and captures the information to instantiate the same reasoning in the absence of a linear programming expert. The figures are screen displays seen by the constituent in all cases. The idea is to have a sense of the form and content of the information in order to understand how the rule-based reasoning functions.

The last section summarizes results obtained in the IMPS project and avenues for further research. The references, which appear at the end, comprise only a small portion of the litera-
ture that has grown rapidly over the past four
years. A more complete bibliography is given in
[18].

1. Terms and concepts

A key word in the title of this paper is “intelligence,” which is sometimes overused in describing a decision support system or an environment for creating one. To avoid confusion, we define this term as it is used in this project. We do not claim to settle the debate over the meaning of intelligence in a broader context.

Presently, the mathematical programming tools available for commercial use are limited to modeling languages with some aids for analysis. All of the reasoning, however, is vested in the persons responsible for building, managing and applying the decision support system that uses linear programming as its main engine. We consider a system to be intelligent if the computer performs reasoning currently performed by people. In addition to mechanisms that include analogical, as well as logical, reasoning, we regard discourse as an important component. In particular, the ability to translate results into English is regarded here as a form of intelligence in itself.

Thus, our meaning of intelligence is based on two things: (1) reasoning by the computer, generally driven by rules, which performs functions currently done by a linear programming expert; and, (2) discourse, which includes natural language explanations of results.

Now consider the syntax of a linear programming model - that is, its rules of composition. One way to express an LP model, which is the way of most textbooks, is first to define sets, domains, data tables and variables, and second to define the constraints. To illustrate, consider the following production-distribution model.

Given:

1. A collection of plants distinguished by their location and processes of operation. Each process requires raw material input and produces finished products.
2. Markets, distinguished by location and product.
3. Transportation links from plants to markets.

Data:

- \( R_{pj} \) = unit amount of raw material i used by process \( p \) at plant \( j \);
- \( Y_{pkj} \) = unit yield of product \( k \) from process \( p \) at plant \( j \);
- \( S_i \) = total supply of raw material i;
- \( COST_{OP_{pj}} \) = unit operation cost of process \( p \) at plant \( j \);
- \( K_j \) = capacity limit of plant \( j \), measured in terms of its total output;
- \( COST_{SH_{jmk}} \) = unit shipping cost of product \( k \) from plant \( j \) to market \( m \), where if there is no link from plant \( j \) to market \( m \) for product \( k \), the value of \( COST_{SH_{jmk}} = \infty \);
- \( D_{mk} \) = demand for product \( k \) required to be satisfied in market \( m \).

Model:

Let \( P_{pj} \) = level of production using process \( p \) at plant \( j \);
Let \( T_{jmk} \) = amount of product \( k \) sent from plant \( j \) to market \( m \).
Let \( COST \) = total cost of production and transportation.

Minimize \( COST \) subject to: \( P_{pj}, T_{jmk} \geq 0 \);
\[
\text{Cost} = \sum_{pj} \text{Cost}_{OP_{pj}} P_{pj} + \sum_{jmk} \text{Cost}_{SH_{jmk}} T_{jmk}
\]

![Fig. 2. A Schema of the production-distribution model.](image-url)
A(i) = ΣρjRρjPρj ≤ S_i (Raw material availabilities)
C(j) = ΣρPρj ≤ K_j (Capacity limits)
B(j, k) = ΣρPρjTρjk - ΣmT_mjk = 0 (Balance equations)
D(m, k) = ΣjT_mjk ≥ D_mk (Demands)

Notice that we began our expression of the model with Let ... Then, we wrote the objective and the constraints. This is the algebraic form. Alternatively, we could express the model in schema form, as shown in fig. 2.

The rows have been classified with COST the objective row, and the other row types begin with A, B, C and D, respectively. The activities have been classified as production (P) and transportation (T). The subscripts in the original problem definition become domains in this expression. Each domain is a cross product of sets. The sets are:

i = raw material;
p = process;
j = plant (location);
k = finished product;
m = market location.

When forming the name of a row or column, its type is distinguished by its first character. Then, its domain is used, but without the parentheses and commas.

For example, consider the transportation activity T(j, m, k) for the particular plant j = DE (a code for Denver), the particular market m = CH (a code for Chicago), and the particular finished product k = T (a code for table). Then, the column name is TDECHT. This syntax is used throughout the examples in subsequent sections.

Suppose, in an instance, we specify the following set elements.

i = {S, W} : S means steel, W means wood;
p = {1, 2, 3} : 1 means process 1, 2 means process 2, 3 means process 3;
j = {N, S} : N means North, S means South;
m = {DE, CH} : DE means Denver, CH means Chicago;
k = {C, T} : C means chair, T means table.

Suppose further that the shipping links are only those shown in fig. 3. Then, the particular linear program is as follows (where ©) simply indicates some data value).

![Diagram of shipping links](image)

Fig. 3. Shipping links for an instance of the production-distribution model (arc labels are products).

minimize COST subject to:

COST = ©P1N + ©P1S + ©P2N + ©P2S + ©P3N + ©P3S + ©TNCHC + ©TSCHT + ©INDEC + ©TDNDET

AS = ©P1N + ©P1S + ©P3N + ©P3S

AW = ©P2N + ©P2S + ©P3N + ©P3S

BNC = ©P1N + ©P3N + ©TNCHC - ©TDNDET

BNT = ©P2N + ©P3N - ©TNDET

BST = ©P2S + ©P3S - ©TSCHT

CN = ©P1N + ©P2N + ©P3N

CS = ©P1S + ©P2S + ©P3S

DCHC = ©TNCHC

DCHT = ©TSCHT

DDEC = ©TNDEC

DDDET = ©TDNDET

All activity levels

The algebraic description is only one view of this instance. Another is to see the sign pattern of the matrix, as pictured in fig. 4. Note that the columns are printed vertically, which is how ANALYZE pictures matrices. Only the signs of the nonzero coefficients and right-hand sides are shown.

For example, consider the first column, P1N. This is the name of activity P(p, j) for p = 1 and j = N. It uses steel, indicated by the + entry in row AS, which is the availability constraint, A(i) for i = S. The activities that produce with process 2, namely P2N and P2S, each use wood; and, those with process 3 use fixed shares of steel and wood. The capacity row, CN, limits the total capacities used by the associated production activities, P1N, P2N and P3N.

2. Price analysis

In this section we illustrate some of the support for an LP expert in order to describe a
Fig. 4. A qualitative view of the LP matrix for an instance of the production-distribution model.

logical approach to interpreting an optimal dual price. Then, we illustrate the automatic interpretation capability.

Fig. 5 illustrates one way to gain such an overview with the SCHEMA and SYNTAX commands. It tells us that there are three activity classes: Supply, Demand and Transportation. It also tells us there are two row classes: Supply balance and Demand balance. The objective is to minimize COST. The row and column classes are defined over domains that are cross products of two sets: materials (MT) and locations (LO). Note that this formulation represents conservation of flows as homogeneous equations; supplies are upper bounds on the supply activities, and demands are fixed values of the demand activities.

To prepare for some analysis, we display the mahogany demand rows in fig. 6. These all begin with DMO, following the model's syntax. As shown in fig. 5, each demand row is D(MT, LO), so for MO a member of set MT and CH a member of set LO, the row name is DMOCH. This is the first row displayed. The first information displayed is the solution status, which is L for each demand row (i.e., it is nonbasic at its lower bound) because the row is an equation. Second, the level of the row activity is zero, as are its lower and upper bounds. Finally, the dual price is displayed, and this is what we want to explain.

Fig. 5. A schema view and the model syntax.
We begin with an exercise to explain the $73 price for the first demand row shown in fig. 6, namely DMOCH. As an LP expert, we might begin, as shown in fig. 7, by picturing the submatrix composed of this row and all basic columns. The PICTURE command removes null rows and columns from the submatrix before giving the picture, so we see the resulting 1×1 submatrix. Among the basic activities, only activity TMOSECH has a nonzero in row DMOCH.

At this point, we may expand the submatrix by bringing in other rows that intersect the one column. This, in turn, may lead to bringing in other columns, preferably basic, until we eventually reach a complete submatrix - that is, where all equations are balanced qualitatively. This has been automated with the TRACE command. Starting with the 1×1 submatrix, TRACE seeks to balance the equations with basic activities. When it is finished, a submatrix is created that contains the original one and identifies the portion of the LP relevant to the flows that determine the marginal price. Then, we picture the resulting submatrix, shown in fig. 8. In this case, the 3×3 submatrix shows the COST and a flow from the supply activity SMOSE1, shipped by the transportation activity TMOSECH, and finally consumed by the demand activity, DMOCH1.

The picture is useful for seeing patterns and having a quick look at a submatrix, but we need the numerical values of the nonzeros to proceed. This is done with the LIST command, as shown in fig. 9.

Of particular importance are the costs: $55 for the supply activity (SMOSE1) and $18 for the transportation activity (TMOSECH). We are now close to an explanation of the demand price, but let us examine the flows.

Fig. 10 shows the result of a display command to see the 3 columns in the submatrix, primarily their levels. The demand activity (DMOCH1) is at its fixed level. The associated transportation activity (TMOSECH) is thus at this level (25), but the supply activity (SMOSE1) level is 75 because it supplies other demands.

You might now have the image of fig. 11 in your mind. We know from the algebraic substructure – that is, the submatrix found – that there is a margin-setting path from supply in Seattle (SE) to demand in Chicago (CH). The supply and
demand boxes may be regarded as the nodes associated with the balance rows (SMOSE and DMOCH, respectively). The arrows represent the 3 activities. The first activity (SMOSE1) supplies mahogany (MO) in Seattle. Its input cost of $18 and its (optimal) level of 75 units are shown above the first arc. The second activity (TMOSECH) transports 25 units of the mahogany from Seattle to Chicago at its COST of $55. The third activity (DMOCH1) consumes the 25 units of mahogany at zero COST. The imputed consumer's cost, however, is $73 per unit, shown below its arc. The row prices, \( P = 18 \) for the Seattle supply balance and \( P = 73 \) for the Chicago demand balance, are shown below their boxes.

We are now in a position to give an algebraic answer to this exercise. The TRACE reveals a path from supply to demand; the levels of the activities in this path may be perturbed to accommodate a change in the right-hand side of the demand row DMOCH (equivalently, the fixed level of the associated demand activity, DMOCH1). The $73 is the total unit cost of this perturbation. The last listing (c.f., fig. 9) shows that the total (input) cost is $55 for the supply activity (SMOSE1) and $18 for the transportation activity (TMOSECH). Because there are no loss or gain factors (i.e., the other coefficients are \( \pm 1 \)), the actual delivered cost is $55 + $18, or $73.

The analysis we just did has been automated. In fig. 12, we use the INTERPRT command, which instantiates a rule file that goes through the same steps, using ANALYZE commands, like TRACE, and interrogating the results in composing a final interpretation.

It is not usually the case that the delivered cost equals the marginal price. We next analyze such a case by performing the same exercise for row DMOLA. The problem is to explain the $68 in terms of the model/solution structure and the data. Fig. 13 starts with the same PICTURE and TRACE specifications for this row as we did for the first row (DMOCH).

Fig. 14 pictures the resulting 3 \times 3 submatrix. So far, the situation looks like the previous case, but let us proceed to examine the flows and costs.

Now we DISPLAY the columns, as shown in fig. 15. We see that the supply activity (SMOSF1) is at its bound (\( S = U \)), which is 30. Upon listing the COST row (using EQUATION format), we see that its delivered cost is only $50. The difference of $50 − $68 is what we seek to explain. This is precisely the reduced cost of the capacitated supply activity, which is displayed as $-18$.

```
ANALYZE ...LIST
Submatrix has 6 NONZEROS
MIN COST = +55 SMOSE1 + 18 TMOSECH
0 = DMOCH = - DMOCH1 + TMOSECH
0 = SMOSE = + SMOSE1 - TMOSECH
```

Fig. 9. Listing the equations in the submatrix.
ANALYZE ... DISPLAY COL
3 Columns are in submatrix.

<table>
<thead>
<tr>
<th>COL</th>
<th>STAT</th>
<th>LEVEL (X)</th>
<th>LO_BOUND</th>
<th>UP_BOUND</th>
<th>PRICE (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMOCCH</td>
<td>L</td>
<td>25.000</td>
<td>25.000</td>
<td>25.000</td>
<td>73.000</td>
</tr>
<tr>
<td>SMOSE1</td>
<td>B</td>
<td>75.000</td>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>TMOSECH</td>
<td>B</td>
<td>25.000</td>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 10. Displaying the columns.

75 @S$55 \rightarrow \text{Seattle} \rightarrow 25 @S$18 \rightarrow \text{Chicago} \rightarrow 25 @S0

\[ D=0 \quad P=55 \quad D=0 \quad P=73 \quad D=73 \]

Fig. 11. A flow diagram corresponding to the submatrix.

ANALYZE ... SUBMAT ROW = DMOCCH
ANALYZE ... INTERPRT PRICE
Row DMOCCH balances demand of mahogany in Chicago. I shall try to interpret its price of 73. The consumers in Chicago receive all of their mahogany from Seattle with delivered cost = 73. Thus, marginal price = delivered cost = 73 (from Seattle) = price of DMOCCH = Supply cost (55) + Transportation cost (18).

Fig. 12. Automatic price interpretation when price = delivered cost.

ANALYZE ... PIC DMOLA,* S=B
Picture of submatrix

T
M
O
S
F
L
A
DMOLA + = 0
ANALYZE ... TRACE
TRACE complete...submatrix is set

Fig. 13. Beginning the price trace for row DMOLA.

ANALYZE ... PIC
Picture of submatrix

D S T
M M M
0 0 0
L S S
A F F
1 1 L
A
COST + + = MIN
DMOLA - + = 0
SMOSF + - = 0

Fig. 14. Picturing the submatrix resulting from the margin-setting path trace.
We must now trace the flow out of this supplier to explain its reduced cost. There is a variety of ways to proceed. We begin to explain the $18 difference by seeing who else receives the other 10 units of supply. Fig. 16 shows the result of picturing the associated supply row (SMOSF) with the columns of not only basic activities, but also of all columns with zero reduced cost. (This distinction will become apparent, as our problem needs to include the dual degeneracy when tracing prices.) As shown at the top of fig. 16, there are 3 columns whose associated activities have \( D = 0 \) and that intersect row SMOSF. We then TRACE this submatrix and picture the result.

We want to see the flows and the reduced costs on the activities in this submatrix. To do so, we DISPLAY the columns, shown in fig. 17. Our story starts to take shape, as the San Francisco supplier ships to other places. Since we know the supply bound (U = 30) is binding, the $18 must
be due to the value of the capacity in fulfilling some other demand(s). That is the story we want to develop.

At this point, it helps to know what the rows and columns mean. Fig. 18 gives their syntax with the EXPLAIN command.

Now we want to see the costs, which we list in fig. 19. This tells us the delivered costs to Denver, LA and Seattle. That is, we already know the supply cost in San Francisco is 45. Now the TRACE and the earlier EXPLAIN tell us that there are 3 activities with zero reduced cost. The first is TMOSFDE, which transports mahogany from San Francisco to Denver at a unit cost of $12. The second is TMOSFLA, which transports mahogany from San Francisco to Los Angeles at a unit cost of $5, which is the one we already saw when we started this exercise. The third is TMOSFSE, which transports mahogany from San Francisco to Seattle at a unit cost of $8.

From fig. 20, we see that only two of these margin-setting flows – that is, where $D = 0$ – are positive. The status of activity TMOSFDE is Basic, but its level is 0. We could explore the link with Seattle, but it is this degeneracy that holds the key for the answer we seek. Fig. 20 pictures the submatrix with the associated demand row (DMODE) and all columns whose activity level is at least 1 (indicated by $X = 1/\ast$). We obtain the answer to the question, “Where does Denver get its mahogany?”

Skipping the new picture after the trace, we immediately list the costs and find two things. First, the mahogany comes from only one supplier, namely Seattle. Second, the delivered cost is $75 ($55 supply, which is activity SMOSE1, and $20 transportation, which is TMOSFSE).

Now we can put the pieces together and construct the analysis story. Fig. 21 shows a flow graph that describes the situation. The San Francisco supplier sends to Los Angeles (where we started) and to Seattle. The latter flow is irrelevant to our original exercise of explaining the $68 marginal price for row DMOLA because it is not the case of the $18 economic rent, which shows up as the $18 reduced cost of the supply activity (SMOSF1). This is due to the margin-setting property in its link with Denver, even though there is no positive flow across that arc (TMOSFDE).

The story is that the demand price for mahogany in Denver, which is $P = 75$ for row

---

**Fig. 17.** Displaying columns in the submatrix.

**Fig. 18.** Explaining the meaning of the columns in the submatrix.
DMODE, is determined as in the first case: it receives its shipment from another supplier, namely from Seattle (via the transportation activity TMOSEDE), with delivered cost of $75. The presence of the link from SF, however, allows displacement if the capacity of the LA supplier would increase. The (actual) cost of shipping from San Francisco to Denver is $57 ($45 supply + $12 transportation), so the savings from displacement would be $18. This is what determines the economic rent we have sought to explain.

There is more than one way to compose the final explanation. One is that the consumer's price of mahogany in LA ($68) is due to demand in Denver and the limited supply in San Francisco. Even if we reduce the production cost and the transportation cost from San Francisco to LA, the price would still be $68 due to the pressure resulting from limited supply capacity. The only way to reduce the $68 price is to reduce the delivered cost from Seattle to Denver!

This example is more complex than the first case because the delivered cost does not equal the marginal price. Fig. 22 shows that the automatic price interpretation is not complete. The analyst, however, is advised to interpret the supply price because once the $18 economic rent is explained, the rest follows. (The term economic rent is used, and this may not be understood by the analyst. It depends on how the model builder (or manager) wrote the rule file.) As advised, we proceed to interpret the supply price with another use of the INTERPRT command (abbreviated I). The result is a successful interpretation, as we did before (but now automatically when we're not there for someone else with the same question).

The complete interpretation may be done with chaining. For this case, we could automatically invoke the interpretation of the San Francisco supply price after the conclusion of the LA demand price. Presently, this is risky for general cases because there is no safeguard against cycling. Unlike the present case, there could be a cyclic structure that cannot be simply decomposed into a causal chain.

We have gone through detail in one particular analysis question to illustrate the automatic inter-
pretation in ANALYZE from rule files that can be prepared by a model builder. In addition, model syntax can be used for translations into English for output formation.

3. Infeasibility diagnosis

In this section we consider the problem of diagnosing the cause of an infeasible instance. Generally, the diagnostic process begins by isolating a portion of the model. Often, more than one form of isolation is a plausible cause, so some reasoning is necessary to explore the alternatives. A diagnosis is an explanation of one of these isolations. A survey of isolation methods that have been used in practice is given by Greenberg and Murphy [21], and an empirical study for blending models is given by Greenberg [19]. Here we use the simplest of those blending models to

```plaintext
ANALYZE ... SUB R DMOLA
ANALYZE ... I P
Row DMOLA balances demand of mahogany in Los Angeles. I shall try to interpret its price of 68. The consumers in Los Angeles receive all of their mahogany from San Francisco with delivered cost = 50 (this supplier, however, has an economic rent of 18). The marginal price (68) = delivered cost + rent. You might try interpreting the supply price... see row SMOSF.
ANALYZE ... SUB R SMOSF
ANALYZE ... I P
Row SMOSF balances supply of mahogany in San Francisco. I shall try to interpret its price of 63. The (input) supply cost = 45 (excluding economic rent = 18). This supply is delivered to 2 consumers -- namely, to Los Angeles with transportation cost = 5; and, to Seattle with transportation cost = 8. This supplier does not send any mahogany to Denver, but it is this consumer that is setting the supplier economic rent. That is, any increase in the supply of mahogany in San Francisco would go to Denver and displace its most expensive current delivered cost.
```

Fig. 22. Automatic price interpretation when price + delivered cost.
Modler ...

LIST MODEL

Model BLEND

Refinery Blending Problem

Minimize COST

Subject to:

\[
\begin{align*}
\text{COST} &= \text{COST\_PU(\text{PU,RF})} \times \text{OP(\text{PU,RF})} + \text{COST\_BU(\text{BU})} \times \text{OB(\text{BU})} \\
\text{SUP(\text{RF})} &= \text{INPUT\_RF(\text{PU,RF})} \times \text{OP(\text{PU,RF})} \leftarrow \text{SUPPLY(\text{RF})} \\
\text{CAP(\text{PU})} &= \text{OP(\text{PU,RF})} \leftarrow \text{CAP\_PU(\text{PU})} \\
\text{BAL(\text{BS})} &= \text{OUTPT\_BS(\text{PU,RF,BS})} \times \text{OP(\text{PU,RF})} - \text{INPUT\_BS(\text{BU,BS})} \times \text{OB(\text{BU})} = 0 \\
\text{DEM(\text{FP})} &= \text{OUTPT\_PU(\text{PU,RF,FP})} \times \text{OP(\text{PU,RF})} + \text{OUTPT\_BU(\text{BU,FP})} \times \text{OB(\text{BU})} \\
&\geq \text{DEMAND(\text{FP})}
\end{align*}
\]

Decision variables:

\[
\begin{align*}
0 &\leq \text{OP} \\
0 &\leq \text{OB} \leq \text{CAP\_BU}
\end{align*}
\]

Fig. 23. Algebraic view of blending model.

illustrate a reasoning for diagnosis formation.

Fig. 23 gives an algebraic view of the blending model, which was created with MODLER, and fig. 24 gives its schema view.

Fig. 25 displays the parameters, sets and tables that comprise the data for this blending model (see [19] for experiments with more general blending models, which are multi-regional and dynamic).

In the following examples we present rules that use the information from an isolation to form a diagnosis RANDMOD [17] was used to generate infeasible instances, based on aggregating some rows and setting the aggregate right-hand sides to be inconsistent with those used in the aggregation.

Three methods of isolation are considered. The first is based on the following slight extension of a basic theorem in linear programming [4].

Theorem. Let \( S = \{ x : L \leq x \leq U, a \leq Ax \leq b \} = \phi; \) and, let \( \pi \) be a phase I dual solution (price vector) associated with the range constraint on \( Ax. \) Then, \( (x : L \leq x \leq U, a \leq \pi Ax \leq b) = \phi, \) where the aggregate range is: \( \alpha = \pi^+ a + \pi^- b, \beta = \pi^+ b + \pi^- a. \)

Notation: \( \pi^+_r = \max[0, \pi_r] \) and \( \pi^-_r = \min[0, \pi_r]. \)

In the experiments, ANALYZE is used to compute and display the aggregate infeasible constraint using prices obtained from MINOS 5.3. When a basic variable is feasible, its aggregate coefficient in \( \pi A \) is the usual reduced cost vector, which is zero. In general, many other coefficients can be zero in fact, it is possible for all aggregate coefficients to be zero, in which case either \( \alpha > 0 \) or \( \beta < 0 \) for the infeasibility.

Typically, there are some nonzero coefficients, which are used for diagnosis formation. A myopic range of \( \pi Ax \) is computed from the bounds, \( L \leq x \leq U; \) and, according to theory, this must lie outside the aggregate range. That is, let \( \lambda = \)

\[
\begin{align*}
\text{MODLER ...SCHEMA DISPLAY} \\
\text{----------: BLOCK SCHEMA :----------} \\
\text{OP(\text{PU,RF})} & \text{ OB(\text{BU})} \\
\text{COST} & \text{COST\_PU} \text{ COST\_BU} \text{ ...MIN} \\
\text{SUP(\text{RF})} & \text{INPUT\_RF} \leftarrow \text{SUPPLY} \\
\text{CAP(\text{PU})} & 1 \leftarrow \text{CAP\_PU} \\
\text{BAL(\text{BS})} & \text{OUTPT\_BS} \text{ -INPUT\_BS} = 0 \\
\text{DEM(\text{FP})} & \text{OUTPT\_PU} \text{ OUTPT\_BU} \geq \text{DEMAND} \\
\text{BOUNDS} & 0 \text{ 0} \\
& * \text{ CAP\_BU}
\end{align*}
\]

Fig. 24. Schema view of blending model.
MODLER ...DISPLAY PARAM,SET,_TABLE
PARAMETER NUM_BU number of blend units
PARAMETER NUM_PU number of primary units
SET PU primary unit = 1,...,NUM_PU
SET BU blend unit = 1,...,NUM_BU
SET RF raw feed stock
SET BS blend stock
SET FP final product
TABLE CAP_BU(BU) capacity of BU
TABLE CAP_PU(PU) capacity of PU
TABLE COST_BU(BU) cost to operate BU
TABLE COST_PU(PU,RF) cost to operate PU using RF
TABLE INPUT_RF(PU,RF) amount of RF used by PU
TABLE INPUT_BS(BU,BS) amount of BS used by BU
TABLE OUTPT_BS(PU,RF,BS) amount of BS produced by PU from RF
TABLE OUTPT_PU(PU,RF,FP) amount of FP produced by PU from RF
TABLE OUTPT_BU(BU,FP) amount of FP produced by BU
TABLE SUPPLY(RF) supply of RF
TABLE DEMAND(FP) demand for FP

Fig. 25. Data information of blending model.

ANALYZE ... SCHEMA DISPLAY *

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

:LO 0 0
:UP * 10/15

DOMAIN INFORMATION

PU  primary unit
BU  blend unit
RF  raw feed stock
BS  blend stock
FP  final product

ANALYZE ... SYNTAX ROW,COL
Row syntax has 4 classes
A row that begins with SUP limits use of some raw feed stock.
A row that begins with CAP limits capacity of some primary unit.
A row that begins with BAL balances some blend stock.
A row that begins with DEM demands some final product.

Column syntax has 2 classes
A column that begins with OP operates some primary unit using some
raw feed stock.
A column that begins with OB operates some blend unit.

Fig. 26. An overview of an infeasible instance of the blending problem.
min[πAx: L ≤ x ≤ U] and μ = max[πAx: L ≤ x ≤ U]. Theoretically, it must be true that either μ < α or λ > β (computational errors could be a problem).

If μ < α, the greatest value of πAx is not enough to satisfy its lower bound, α. In this case, the activities with positive coefficients have upper bounds that are too low or the activities with negative coefficients have lower bounds that are too great. The first class suggests not enough capacity or raw feed stock, like crude oil. The second class suggests too much demand or contract requirements on operations. The diagnosis, then, is to present these facts with other relevant information about the activities.

If λ > β, the least value of πAx is too much to satisfy its upper bound, β. In this case, we form the activity classes as before with a similar inference. Those with positive coefficients have lower bounds that are too great, and those with negative coefficients have upper bounds that are too low.

Let us illustrate how this aggregate infeasibility can be used to give a diagnosis. Fig. 26 gives an overview of the model structure and this instance. The block schema and the syntax for the rows and columns are displayed by the associated ANALYZE commands.

We see there are two classes of activities, which comprise the columns of the LP matrix: OP and OB, which operate primary and blend units, respectively. There are four classes of constraints, which, together with the COST, comprise the rows of the LP matrix: SUP limits the use of raw feed stocks by their supplies; CAP limits the operations of the primary units by their capacities; BAL balances blend stocks, which are outputs of primary units and inputs to blend units; and, DEM requires demand for final products to be satisfied, perhaps with surplus.

Fig. 27 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, 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.

Inserting the meaning of the activities and their upper bounds, this leads to an immediate diagnosis:

There is insufficient capacity to operate blend units 01 and 04 to satisfy the constraints. In particular, their capacities, say U01 and U04, respectively, must satisfy:

0.8055 U01 + 0.7911 U04 ≥ 32.373,

but their capacities, U01 = 10 and U04 = 10, yield only 15.966.

Putting the English text aside, the essential feature of the underlying rule is that we found positive coefficients with positive (finite) upper bounds for the two activities. This enabled the diagnosis.

The diagnosis obtained from the Phase I aggregation method of isolation does not answer the question of why this aggregate constraint must hold. That could be all right in some cases, but other cases require deeper reasoning in the diagnosis. In particular, it may be desirable (perhaps necessary) to give a diagnostic purely in terms of the original coefficients. A method of isolation that does this is successive bounding, and it is executed by the REDUCE command in ANALYZE.

Successive bounding, which has an early history in pre-processing linear programs for a variety of purposes, applies simple tests iteratively to infer tighter bounds on the activities and on the range constraints. In some cases, the method
detects infeasibility, possibly with other reductions that are irrelevant to its cause.

If infeasibility is detected, it is with a range constraint for an equation, perhaps after some of the other bounds have been tightened. A diagnosis is obtained by finding a causal chain of bound tightening that led to the detection. For the infeasible instance shown in fig. 26, successive bounding enabled the following diagnosis to be formed.

Blend stock heavy naphtha cannot be balanced due to the following.
1. OB01 <= 10 and OB03 <= 10 and OB04 <= 10 and Row DEMPG imply OB02 >= 0.6667.
2. Row CAP04 implies OP04MC <= 10.
3. OP04MC <= 10 and OB01 <= 10 and OB04 <= 10 and Row BALLN imply OP05MC <= 8.
4. OP04MC <= 10 and OP05MC <= 8 and OB02 >= 0.6667 and Row BALLN imply OB01 <= 6.19.
5. OP04MC <= 10 and OP05MC <= 8 and OB02 >= 0.6667 and Row BALLN imply OB02 <= 6.6857.
6. OB02 <= 6.6857 and OB03 <= 10 and OB04 <= 10 and Row DEMPG imply OB01 <= 1.81.
7. OB01 <= 1.619 and OB03 <= 10 and OB04 <= 10 and Row DEMPG imply OB02 <= 3.206.
8. OB01 <= 6.19 and OB02 <= 6.857 and OB04 <= 10 and Row DEMPG imply OB03 <= 9.262.
9. OB02 <= 6.857 and OB03 <= 10 and OB04 <= 10 and Row DEMPG imply OB04 >= 6.349.

10. OB01 >= 1.81 and OB04 >= 6.349 and OP04MC <= 10 and Row BALLN imply OP05MC <= 5.714.

Now the range of row BALLN is determined with these bounds as:

Activity Coefficient Bounds Range contribution
OB01 -0.7 1.81, 6.19 -4.333, -1.267
OB02 -0.7 3.206, 6.857 -4.8, -2.244
OB04 -0.2 6.349, 10 -2, -1.27
OP04MC 0.4 0, 10 0, 4
OP05MC 0.1 0, 5.714 0, 0.5714

Thus, max of BALLN = -1.267 -2.244 -1.27 +4 +0.5714 = -0.2096, so this balance equation cannot be satisfied.

The causal chain (1 – 10) was obtained from the REDUCE log, which contained 10 irrelevant reductions, plus the relevant equations, listed in fig. 28. The general structure of the diagnosis is to begin 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 BALLN. Then, the causal chain is given. In this example we merely used the names of the variables for each bound tightening, but we could have translated each activity using the syntax. The last part of the diagnosis is

---

```
ANALYZE ... LIST
   0 = BALLN = -0.7 OB01 -0.7 OB02
   -0.2 OB04 +0.4 OP04MC
   +0.1 OP05MC

   0 = BALLN = -0.3 OB01 -0.1 OB04
   +0.1 OP04MC +0.5 OP05MC

   10 => CAP04 = + OP04MC + OP04MC

   10 <= DEMPG = +0.2 OB01 +0.3 OB02
   +0.4 OB03 +0.3 OB04

   15 <= DEMRC = +0.6 OB01 +0.6 OB02
   +0.4 OB03 +0.5 OB04
```

Fig. 28. Listing the relevant equations inferred from the REDUCE log.

```
ANALYZE ... SCH D T
  1:0
  1:0

OP(PU,RF) OB(BU) RHSMODL
SUP(RF) 1 20
CAP(PU) 1 20
BAL(BS) 0.05/0.6 -1/-0.1 = 0
DEM(FP) 0.1/0.5 0.1/1 >= 5/15
INFEAS01 -1 <= .41
COST ...MIN
:LO 0 0
:UP * 20
```

Fig. 29. Schema for another infeasible instance of the blending problem.
Fig. 30. The phase I aggregation of the supply instance.

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.

Successive bounding can fail, and it is very unpredictable. When it does detect infeasibility, and when the causal chain is discernable (i.e., not confounded by a large number of irrelevant reductions), its information enables a deeper diagnosis than Phase I aggregation.

Phase I aggregation can also fail. We illustrate one way with another infeasible instance of the same problem, shown in Fig. 29.

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, as shown in Fig. 30, for this kind of infeasibility. This is because all aggregate coefficients must be zero when a row is a linear combination of other rows.

The rule to use the information from Phase I aggregation does not fire due to having no non-zero coefficients to explain. Successive bounding failed to detect infeasibility in this case, so we turn to another method of isolation.

Another method 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 simple non-negativity, are included in the formation of this set. Originally suggested by van Loon [27], the most extensive algorithmic development of this approach has been by Chinneck [2] (see, also, [3]). An irreducible infeasible subsystem (IIS) is computed outside of ANALYZE (using Chinneck's code) and read into memory during the session. From a software view, this is integrated under a windows environment, but it is presently necessary to have the user execute the IIS code.

For the example in Fig. 29, the IIS is obtained and shown in Fig. 31. To form a diagnosis, one could have the rule file simply give this information in the form shown, or with English translations, and add text to tell the user that this portion of the problem cannot be satisfied. Mathematically, the IIS just obtains the isolation; it does not go further in finding out that row INFEAS01 is a linear combination of the other rows in the submatrix.

Although they 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. Thus, we view these three methods of isolation as engines, and the rule-base

Fig. 31. Listing the equations of the IIS rows for the infeasible instance in Fig. 29.
is responsible for deciding which engine to use and how to interpret the information for the final diagnosis formation. The following illustrates what the ANALYZE rule-base might contain.

Rule INFEAS. Execute the AGGREGATE command.
IF some coefficient $< 0$ THEN fire Rule AGGREGATE
ELSE Execute the REDUCE command and IF detection succeeds THEN fire Rule REDUCE
ELSE Execute IIS and fire Rule IIS.

The three rules, AGGREGATE, REDUCE and IIS, are interpretations of the information obtained from the associated engines. These could be constructed to be general or particular to the model. The former requires less maintenance, and the latter can be worded sharper.

4. Reasoning about redundancy

A constraint is redundant if its removal does not change the set of feasible solutions. There are several ways to detect redundancy, but our interest is in explaining the reason for the redundancy. There are at least two reasons for this need to know why a constraint is redundant. First, we can deepen our understanding of the solution by realizing this does not constrain any solutions. Second, we can aid model management, which can include model revision.

The last analysis problem we now consider is a question of redundancy for an optimal instance of the same blending model used in the previous section. Fig. 32 shows a display of two activities and an English translation of their meaning. Our task is to explain why the non-negativities of O0221 and TUP211 are redundant. The primary objective here is to better understand the meaning of the solution. In particular, the redundancies show that the operation and transportation levels must be positive, regardless of costs. In some cases such redundancies are expected, but in other cases they could reveal an error in the model or in the data, so redundancy detection alone can be a model management aid.

We intend to go beyond the detection and explain why these redundancies occur in this in-
stance. Fig. 33 starts with the simplest of the redundancies, namely the non-negativity of TUP211, by picturing it with its equations.

It now is evident that the processing units in Louisiana (region 1) do not make unleaded premium, so the redundancy is due to the demand requirement. The following explanation can be automated by using the ANALYZE commands, including syntactic translation into English, with a rule file.

```
ANALYZE ... PIC
Picture of submatrix

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ......
Activity TUP211 transports unleaded premium from Louisiana to Texas at 1st time period. Its level is required to be at least 0, but this is redundant due to the following reasons. The activity fulfills demand for unleaded premium in Louisiana, and no other activity can fulfill this demand, which is strictly positive.

The redundancy of the operation activity, O0221, is not as obvious. We begin with a picture of its non-zero pattern, as before, shown in fig. 34. Unlike the simple case, this does not reveal why its level must be positive in every feasible solution of this instance, since there are other operation activities with a positive entry in each of its balance rows. We must therefore use other information.

The appropriate ANALYZE command that gives the information we need is the RATEOF command, which shows the rates of substitution for activity O0221. This is a re-write of the original equations that reveals how its level relates to the levels of the nonbasic variables (although the partitioning of basic versus non-basic is particular to this solution, the re-write of the equations is valid in any case). Fig. 35 displays the rates in equation format (specified by $\|$E$|$), followed by displays of the relevant bounds.

This information can be obtained from within a rule file and the following explanation automatically composed.

Activity O0221 operates 2nd primary unit in Louisiana at 1st time period. Its level is required to be at least 0, but this is redundant due to the following reasons. The activity's level satisfies the following equation (by implication of the original equations).

\[ 00221 = BUP11 + BUP21 + IUP11 + IUP21 - 01321, \]

where

- BUP11 balances unleaded premium in Louisiana at 1st time period $\geq 100$;
- BUP21 balances unleaded premium in Texas at 1st time period $\geq 150$;
- IUP11 inventories unleaded premium in Louisiana at 1st time period $\geq 0$;
- IUP21 inventories unleaded premium in Texas at 1st time period $\geq 0$;
- 01321 operates 13th blend unit in Louisiana at 1st time period $\leq 200$.

These relations imply that the level of O0221 must be at least 50.

The rule file could vary the wording and general form of response. Further, one might test for whether a redundancy depends on the particular data, such as this one above, or is structural, like the first one. In the case of a structural redundancy (i.e., one that depends on the sign pattern, but not the particular numerical values for the instance), the explanation could result in a model revision. In either case, a redundancy explanation offers an aid for model management.

5. Summary and conclusions

The analysis component of the Intelligent Mathematical Programming System, called ANALYZE, is an open architecture that allows users to specify rule files that are instantiated with its INTERPRT command. Applications that have been demonstrated include explanations of dual prices, diagnoses of infeasibilities, and suggestive reasoning about redundancy (other examples appear elsewhere, such as in the primer).

Fig. 36 shows how the ANALYZE design separates engines from cognitrons. In addition to reasoning mechanisms, which use a variety of engines, some internal and some external to ANALYZE, the model syntax enables English translations of the rows and columns that define the linear program as well as the sets in their domains.
Although this paper focused on this rule-base capability and the syntax-driven English translations, the IMPS contains other modules. Different forms of intelligence are explained elsewhere (e.g., [7,20,14]). There are, however, many avenues for research. The following summarizes what we have learned in this project and where we are heading in building a more complete IMPS for future generations.

Formulation

Relational concepts. This includes conventional database theory, automatic linking, and structures that unify functional equations and logical expressions. Object relations, fundamental both to modeling and to intelligence, are represented by conceptual graphs.

Blocking and pattern-matching. This offers an important paradigm useful for model management and analysis. Many questions can be simplified by first blocking and matching patterns over the associated condensed graph. In addition, model formulation is simplified by providing one form of assistance for block construction, which are micro-views, using pattern matching of problem features. Another form of assistance is using syntax to link the blocks automatically.

Prototyping. During early stages of formulation, prototyping may be used to fill in missing data in order to test model robustness, as in MODLER. Data checks can be automated for downstream model management. For example, to prevent an infeasibility, a partial sum of supplies or capacities may be required to be at least as great as an associated partial sum of demands; and, a partial sum of costs may be required to be positive. This has some ad hoc experiences in the field, and we have learned how to incorporate this effectively into the IMPS.

Structured modeling and graph inversion. As approaches to assisting the modeling process, these begin to transfer some of the art to the realm of science, but these theories remain untested at present. The former (developed by Arthur Geoffrion) has advanced to the stage of a linguistic implementation [5,6]; the latter (developed by Harvey Greenberg, Richard Lundgren and John Maybee [20]) has recently provided the basis for a graphics interface capable of testing for qualitative consistency.

Knowledge output. This pertains to new responsibilities for the formulation module, replacing the old notion of matrix generators. We have a better understanding of what should be the form and content of the knowledge output to support subsequent management and analysis.

Analysis

Sensitivity analysis. We have gone beyond the quantitative calculation of rates and ranges to support What if...?, Why..., and Why not...? queries. Qualitative sensitivity analysis plays a greater role, such as in path tracing to explain equilibrium prices and quantities.

Substructures. Recognition algorithms for embedded netforms and other special structures have been developed to support analysis in new ways. Previously motivated to recognize substructures in order to speed up optimization, these heuristics now help to organize model components in a way that simplifies its explanations, as for documentation or scenario analysis.

Rulebases. Rulebase designs have gone beyond the purely logical-based reasoning, drawing from years of analysis experience with a variety of problem and model types. Expert approaches may be captured to some extent in the ANALYZE system, but the scope of this approach is a primary target for further study.

Infeasibility diagnosis. Large, complex models, especially when eclectic, are prone to errors, and they are the most difficult to isolate. New approaches have been developed, and some have been implemented. Simple checks catch some of the common errors in practice, but the more sophisticated methods are needed in some cases.

Reformulation and knowledgebases. The knowledge output of the Model Assistant may determine effective reformulations. In some cases, such as in nonlinear and combinatorial structures, this may be to simplify the model for optimization. In other cases, reformulation may contribute new knowledge to the overall knowledge base. Triggers for reformulation may be purely analytical or may be based on repeated problem-solving (either
real or while prototyping). In other cases, bound reduction techniques offer information for model simplification by reformulating blocks and their substructures.

Discourse

Syntax. Models follow rules of grammar, at least implicitly, which can be formalized, integrated, and exploited for interface and intelligent support.

Semantics. Many semantic forms are possible. Some are discipline-oriented, like economic notions of inputs and outputs; others are structure-oriented, like linked blocks.

Iconic-based graphics. Graphic interfaces are emerging with the advances in technology of hardware and movements toward standardization. Beyond interfaces, graphics offer new ways to incorporate intelligence through deeper discourse using shape (as well as color and position) to carry information in problem domains.

Associative mappings. Discourse association pertains to mappings among text, graphics and algebra. A model assistant could use associative mappings to match problem domain features with portions of a model in a library. Analysis can use associative mappings to guide a heuristic search for infeasibility diagnosis or direct sensitivity queries: What if...? Why...? and Why not...? These same associative mappings, which may be represented by neural networks, could separate text, graphics and algebra. The first may be for automatic documentation; the second may be for analysis support; and, the third may be for standard file transfers to optimizers.

Views. Different people respond differently to the many views of models and instances, and the cognitive aspects are largely uncharted. The traditional view in LP stems from integrating matrix generation and report writing. The recent developments of algebraic languages fulfill a need to provide another view, particularly with the incorporation of relational algebra over sets for restricted domain specifications and conditional generation that is part of the model's logic. More recently, several researchers have developed other views: schema, linked blocks, process networks, entity relationship, structured modeling, net-}

forms, and fundamental graphs. An in-depth analysis of these views is in [22].

Integration

Discourse, formulation and analysis. This triad, which comprise the IMPS foundation, requires integration in at least two ways. First, explicit integration is the essentially obvious thread, such as using the same icons and graph grammars for analysis as for formulation. Second, implicit, or learned, integration is like a cross-referencing in the knowledge bases. How they are linked and their consistency maintained are targets for further study, but we now have articulated what we believe are the right questions.

Knowledge base designs. The form and content of knowledge bases have been the subject of study from many angles, ranging from very general to very specific. The IMPS approach is one of distributed knowledge bases, linked like a hypertext frame by a master knowledge base. The theory of such a design – both in form and in content – is only partially understood, and implementation poses several problems to be addressed.

Learning models. Rule-based learning is included, but it comprises only a portion of the learning models we have investigated. Analogue reasoning has more potential for some of the intelligence we intend to develop.

Elements of implementation. All components have been implemented with real, large models in mind. Although developed for research goals, ANALYZE is already used in practice. Some module implementations are clearly prototypic, but most are sophisticated enough to test with real problems, some provided by consortium members.

Acknowledgements

The author gratefully acknowledges support provided by a consortium of companies: Amoco Oil Company, Shell Development Company, Chesapeake Decision Sciences, Inc., GAMS Development Corp., Ketron Management Science, MathPro, Inc., and Optimum Methods, Inc. Additional support for the development of ANA-
LYZE was provided by General Research Corporation, US West, and the Office of Naval Research (contract No. N-00014-88-K-0104). Additional technical advice was given by Frederic H. Murphy and an anonymous referee.

References

[17] H.J. Greenberg, A Primer for RANDMOD (Mathematics Department, University of Colorado, Denver, CO, 1990c).