The ANALYZE rulebase for supporting LP analysis

Harvey J. Greenberg

Mathematics Department, Campus Box 170,
University of Colorado at Denver, PO Box 173364,
Denver, CO 80217-3364, USA

E-mail: hgreenbe@carbon.cudenver.edu

This paper describes how to design rules to support linear programming analysis in three functional categories: postoptimal sensitivity, debugging, and model management. The ANALYZE system is used to illustrate the behavior of the rules with a variety of examples. Postoptimal sensitivity analysis answers not only the paradigm What if ...? question, but also the more frequently asked Why ...? question. The latter is static, asking why some solution value is what it is, or why it is not something else. The former is dynamic, asking how the solution changes if some element is changed. Debugging can mean a variety of things; here the focus is on diagnosing an infeasible instance. Model management includes documentation, verification, and validation. Rules are illustrated to provide support in each of these related functions, including some that require reasoning about the linear program's structure. Another model management function is to conduct a periodic review, with one of the goals being to simplify the model, if possible. The last illustration is how to test new rule files, where there is a variety of ways to communicate a result to someone who is not expert in linear programming.

Keywords: Linear programming, large-scale systems, computer-assisted analysis, computational economics, sensitivity analysis, model management.

1 Introduction

This describes an approach to provide an intelligent computing environment for analyzing results of a linear program (LP), which has been incorporated into a software system called ANALYZE [9,22]. The types of analyses described here fall into three functional categories: postoptimal sensitivity, debugging, and model management.

The approach was introduced by the author [6-8] in 1977 at the U.S. Department of Energy and has evolved since then with a variety of applications in both the public
and private sectors. Some of the mathematical methodology, particularly for post-optimal sensitivity analysis, is much older, some has been developed over the past fifteen years, and some is new. In addition to the mathematics, however, the idea of intelligence means having the computer perform reasoning functions that are comparable to what is done by people who are expert in LP. In addition to the reasoning, the results must be communicated to a non-expert (in LP). This can be done with English text and/or graphical displays that are composed from problem-domain information.

In general, reasoning mechanisms can be logical or analogical. In this paper, and in the ANALYZE implementation, all reasoning is logical. The mechanism is rule-driven, as in an expert system, and the issue is how to write the rules. Besides ANALYZE, MIMIE [1,2] is a software system that enables an expert to write rules that support analysis by others, when the expert is not present. Some other systems provide extensive discourse models, including graphics (see Jones [37] for an excellent review).

In addition to the references given here, there are hundreds more that are relevant to the creation of an artificially intelligent environment for analysis support. The bibliography [30] given elsewhere in this volume provides a fairly extensive list, with a cross reference index to analysis and discourse (among other, related aspects).

The rest of this paper is organized as follows. Section 2 gives an overview of the ANALYZE system used to illustrate the rule-based analysis support in the rest of the paper.

Section 3 describes rules for postoptimal sensitivity analysis (for background, see [24,25,31]). The paradigm What if ...? question is considered after the more frequently asked Why ...? and Why not ...? questions. The question of why an LP solution has a particular result is considered static because it pertains to the meaning of the particular solution without changing anything. Such questions probe into why the LP result occurred, so the analyst can better understand the reason for its optimality, then relate this to the system represented by the LP model. The question of what if something changes is considered dynamic, and it contributes insight into the particular solution as well as understanding what would happen in response to some change.

Section 4 describes rules for debugging an LP. The focus is on a particular result, but the methodology also applies to model debugging, particularly if the data used is for prototyping. Anomalous results can be treated as part of the debugging process, or it can be included in the approaches described in section 3. Infeasible results (primal or dual) are squarely in the debugging category, which is the focus of this section (for background, see [10,12,15,18,34]).

Section 5 describes rules for model management. This includes automatic documentation, verification and validation. Documentation means reporting what is in the LP in terms that can be understood by users of the model, not necessarily expert in LP, as well as the experts that manage the model and/or its data. Verification is
confirming that what is in the LP is what is believed to be there, and this includes implied relations, which are inferred automatically. Validation pertains to how well the LP represents the system in the context of its value for decision support. Another model management function is simplification. This can occur by periodic review, leading to dimensional reductions and/or to structural simplification. An example of the latter is to discover an equivalent network model that is easier to solve, use, and explain.

I then conclude with some summary statements about the present state of intelligent analysis support and its near-term future. An appendix is included to give some particulars about ANALYZE that are not described in the text.

2 Overview of ANALYZE

The purpose of ANALYZE is to provide computer assistance for analysis (see [22] for a comprehensive description; succinct introductions are in [9,13,20]). It is presumed that a linear program has already been formulated, and an instance has been generated with some language. In general, there are three levels of using ANALYZE, each affecting its successor level.

At the lowest level, ANALYZE provides convenient interactive query to navigate through an LP, perhaps with a solution already obtained from some solver (a solution is not necessary for productive use of ANALYZE). One of the key features is its multi-view architecture [35].

A second level of use provides procedures to assist analysis in a variety of ways. Standard sensitivity questions, such as What if...?, Why...? and Why not...?, can be answered with easy access to information about the solution, including implicit information obtained from a variety of procedures. In addition, diagnostic analysis, such as when the LP is infeasible, can be resolved efficiently using the same procedures. The third level provides an artificially intelligent environment, including English translations of results automatically from the LP syntax [11,28], which is the focus here (for more background, see [14,16,19,21]).

Figure 1 shows an input/output schematic of ANALYZE. Among the input files listed, the only one required by ANALYZE is a matrix file. This is the (de facto) standard MPS file, which is accepted by all commercial optimizers and can be generated by most LP modeling languages. For model analysis this may be sufficient, but other files are useful, such as a solution file from an optimizer. (Matrix and solution information can enter ANALYZE in other ways, avoiding the computationally expensive creation of matrix files and formatted read statements.)

The ANALYZE commands are listed in the appendix. I shall use some of them in the examples, but I shall emphasize the basic concepts that underlie their use, particularly in creating rules that support analysis. In order not to obscure these basic concepts, the actual ANALYZE commands will not appear in the figures that show the results that the user sees. I shall, however, say more about how such rules are put into files.
A rule file is a way to use ANALYZE with responses to questions of analysis. The INTERPRT command in ANALYZE is what the user specifies to instantiate a rule file, and that is what I shall focus on throughout this paper. It contains text, references to the LP information, and special commands that include conditional branching. Text appears in free format until LP information is needed. Then, there are two ways to insert information about the LP into the response that the user sees (or use that information to decide what to do next in the rule file).

The first way is with a simple keyword lookup. The keywords (listed in the appendix) can be character, integer, real, or logical values. When referencing a keyword in the rule file, a percent sign (%) precedes its name. For example, consider the following template:

Problem %PROBLEM is %STATUS, with objective to %OPT %OBJ.

The keywords are PROBLEM, STATUS, OPT, and OBJ.

Suppose the problem name is WOODNET, the solution status is OPTIMAL, the sense of optimization is MINIMIZE, and the name of the objective row is COST. Then, the user will see the following sentence:

Problem WOODNET is OPTIMAL, with objective to MINIMIZE COST.

Text continues in this fashion, substituting values of keywords, until a rulebase command is specified. (These commands are listed in the appendix.) Text buffering is managed by ANALYZE, with screen dimensions and format options under the user’s control, or controlled from within the rule file.
In addition to the system keywords, the rule file can specify its own parameters. These are referenced in the rule file's text with a double percent sign (\%\%). Figure 2 gives an example of what the user sees when instantiating a rule file to give an overview of an LP that was just read into memory.

```
ANALYZE ... INTERPRT MODEL
WOODNET is a network model with 18 production activities, 20 consumption activities, and 55 transfer activities. Its syntax is shown by the following block schematic.

S(MT,SR) T(MT,SR,DR) D(MT,DR)
S(MT,SR) 1 -1 = 0
D(MT,DR) 1 -1 = 0
COST 10/60 5/20 ...
:LO 0 0 5/100
:UP 5/* *

DOMAIN INFORMATION
MT material
SR location
DR location

The row and column classes have the following meanings.
A row that begins with S balances supply of some material in some location.
A row that begins with D balances demand of some material in some location.
A column that begins with S supplies some material at some location.
A column that begins with T transports some material from some location to some location.
A column that begins with D demands some material at some location.
The resident solution is OPTIMAL with COST = 25135.
```

Figure 2. Overview of WOODNET model given by instantiating a rule file named MODEL.

The first paragraph was obtained by executing an ANALYZE procedure that obtains a maximal embedded network. In this case, the entire LP is a network, which is ascertained by having obtained the numbers of rows and columns in each category (a production activity has a single positive coefficient; a consumption activity has a single negative coefficient; and a transfer activity has one positive and one negative coefficient). The paragraph is a template in the rule file, using keywords for the numbers. The last sentence is also a template, and the rest of the response is the direct output of ANALYZE commands.
**ANALYZE ... INTERPRRT MODEL STATS**
The resident solution of WOODNET is OPTIMAL with MIN COST = 25315.
It has the following distribution of solution values.

There are 20 equations that balances supply of some material in some location. All of these have right-hand side = 0 and positive prices.

There are 15 equations that balances demand of some material in some location. All of these have right-hand side = 0 and positive prices.

There are 18 activities that supplies some material in some location. All of these have lower bound = 0, and 14 have upper bounds. 6 activities in this class are setting marginal prices (basic), but there is one degeneracy (its level is zero). 2 activities are at their lower bounds, but one is degenerate (its reduced cost is zero). The remaining 10 activities are at their upper bounds.

There are 55 activities that transports some material from some location to some location. All of these have lower bound = 0 and no upper bound. 29 activities in this class are setting marginal prices (basic), but there are 2 degeneracies (their levels are zero). 26 activities are at their lower bounds, but one is degenerate (its reduced cost is zero).

There are 20 activities that demands some material in some location. All of these have fixed, positive levels, and all of their reduced costs are positive.

Figure 3. Solution statistics of WOODNET instance given by instantiating the MODEL rule file with STATS specified as a special parameter.

There are special parameters, named 1, 2, ..., 9, as in a DOS bat file, which can be passed from the user’s INTERPRRT command specification. Figure 3 shows what the user sees when specifying STATS as parameter 1. Parameters are only one way a user can pass information or specifications to the rule file; if STATS had not been specified, the rule file would give a more complete model interpretation, but the rulebase designer could also prompt the user for information. As in most rule files, this contains templates, using keywords for some of the information. It also has branching logic to compose the right sentence, and it uses looping to gather some of the statistics with its own parameters.

### 3 Rules for postoptimal sensitivity analysis

There is a variety of queries that help to deepen one’s understanding of a solution. In this section I consider two types: *Why...?* and *What if...?*. The *why* questions ask something about why the solution is what it is, and I regard it as static because nothing is changed. The *what if* questions ask something about what the solution response would be if something is changed, so I regard it as dynamic.
3.1 Static/Why?

I begin with questions of the form: Why is the price of <some row or column> equal to <its solution value>? One can refer to the recent tutorial series [24–27] for some of what I shall do, but here I focus on building rules.

The first part of the rule file checks a few simple things, such as the relevance of the resident LP (for example, ensuring it is optimal). If a row has surplus (or slack), its dual price is zero. In this case, the text can be simply generic:

This row has surplus, so its marginal price = 0. This means it does not contribute to the cost or revenue of any activity. Instead, it behaves as though the constraint is absent.

Notice that the response does not say anything about a perturbation of the right-hand side. This is because the user’s query is static, so a dynamic response could be confusing, rather than enlightening.

More commonly, the dual price is positive, and the goal is to explain its value. If the resident solution is basic, we can compute how each basic variable’s level must change in response to some perturbation (this probe is hidden from the user). This is the classical approach, and there are several shortcomings.

First, the basis might not remain feasible for the perturbation. We could interrogate alternative optimal bases to find one that is compatible with the perturbation. If there is one, the resident solution changes, so it becomes important to understand the context of the query, particularly its connection with other queries. Changing to an alternative optimum might become misleading in an overall analysis context.

Second, it could also be that the LP becomes infeasible. For example, suppose we have a pure transportation problem where total supply equals total demand. Then, we cannot decrease one supply or increase one demand; it is necessary for this to be joint changes such that total supply is at least as great as total demand. This means the interpretation of a supply or demand price is not necessarily best understood in terms of perturbation.

Third, even if the rates are mathematically correct, presenting them as the answer might not give enough insight to the user, especially since the question is static, and the user might not understand its connection with perturbation. Another approach is to seek a sequence of responses in terms of the original matrix values. One such approach is path tracing, which uses the sign pattern in the LP matrix to infer directions of changes that must occur. Although it is inspired by network models, it applies to any LP. Often, this provides a first step towards understanding the price in question.

Let us illustrate with several possible responses (using different rules) for the WOODNET LP. Then, I shall consider another example and show other ways to explain prices. Figure 4 shows a table of demand prices, and I begin with the price of mahogany (MT = MO) in Chicago (DR = CH), which is the dual price of $73 for row DMOCH.
Figure 4. Demand prices for the WOODNET LP.

<table>
<thead>
<tr>
<th>MT</th>
<th>CH</th>
<th>DE</th>
<th>LA</th>
<th>SE</th>
<th>SF</th>
</tr>
</thead>
<tbody>
<tr>
<td>MN</td>
<td>73</td>
<td>75</td>
<td>68</td>
<td>71</td>
<td>65</td>
</tr>
<tr>
<td>OK</td>
<td>60</td>
<td>69</td>
<td>62</td>
<td>65</td>
<td>60</td>
</tr>
<tr>
<td>PI</td>
<td>22</td>
<td>22</td>
<td>17</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>WA</td>
<td>78</td>
<td>68</td>
<td>75</td>
<td>78</td>
<td>70</td>
</tr>
</tbody>
</table>

Figure 5. Instantiating a rule file to interpret the price of row DMOCH in WOODNET.

ANALYZE ... INTERPRRT PRICE DMOCH
Row DMOCH 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 DMOCH = Supply cost ($55) + Transportation cost ($18).

Figure 5 shows the response from the particular rule file that is instantiated. This is a particularly simple case because the delivered cost equals the marginal price. Using path tracing, which is automatic in ANALYZE, the rule file determines this. One activity supplies all of the mahogany demand in Chicago, and the two activities in this path (namely, SMOSE and TMOSECH) account for the marginal price, as stated.

The next example is not as simple because the delivered cost does not equal the marginal price. The first paragraph in figure 6 gives the first part of the interpretation using the same rule file. It gives all the information, taken from the margin-setting path obtained by the same logic as in the previous example, but here the supply activity is at its upper bound. Its reduced cost corresponds to an economic rent for the supplier, which is what remains to be interpreted. This rule automatically chains to interpret the supply price (row SMOSF).

Figure 7 gives another view of the information used by the rule file. Instead of English sentences, this graph could just as well have been presented. It shows the relevant portion of the LP, composed of the adjacent margin-setting paths. To trace the flow and associated prices, begin with our starting point: row DMOLA. This is the demand for mahogany in Los Angeles, and its box is in the Northeast corner of the diagram.

The diagram shows that all 20 units demanded in Los Angeles come from San Francisco. The consumer's price is the sum of the supplier's price and the transportation cost: $P = 63 + 5. The $5 transportation cost is direct (data), but the $63 supplier's
price is the sum of its direct cost ($45) and what I called the "economic rent" ($18). The economic rent shows up as the negative reduced cost on the associated supply activity, and our interpretation will become complete once we can explain it.

The diagram shows positive flow to one other consumer, namely in Seattle. We could explore that to see if it explains the rent, but it is a dead end. Its consumer price
(§71) is also an effect, not a cause, of the rent. The key is the degenerate flow to Denver, and the rest of the story is given in the English response in figure 6.

So far, we have the following steps, which contain generic rules – that is, that do not depend upon knowing the LP we are analyzing:

1. Check relevance and simple interpretations, such as when the row has surplus in the solution.

2. Seek a causal chain, such as using path tracing, that explains the relevant trade-offs.

3. Instantiate another rule file.

We proceed to the next step when failure occurs, and the steps are organized by increasing order of complexity.

If the generic rules in steps (1) and (2) fail, classes of LPs can be considered in step (3). If the resident LP is one of the recognized classes, control transfers to a more specialized rule file that uses properties in the class. This can be progressively refined, ending with a directory of LP models. If the resident LP is one of those, a very specialized rule file can be instantiated, which exploits properties of the particular LP model. (I use the term “model” to mean its symbolic form, reflected by its syntax, not just the specific instance. Thus, it is not suggested that a rule file be created for each possible scenario.) This approach leads to a hierarchical rulebase design, with the generic rules at the root and progressively more specialized rules at the deeper levels.

Figure 8 shows another example, which determines electricity generation from each of three fuels: coal, oil and uranium. Because it is so small, the full LP is shown, and each row and column is explained, using the syntax that is read by ANALYZE (from a syntax file).

The optimal tableau is shown in figure 9. The dual price of row DEL is $52, and a user wants to know why.

If we apply the same rules as for the WOODNET example (with some changes in the templates used), we would obtain what is shown in figure 10. If the user asks, Why not use coal or uranium?, different rules apply.

In the case of asking why not use uranium, the rule finds a blocked path – that is, there is already some electricity generated from uranium, and there cannot be any more due to the capacity limit. The computer’s response, therefore, is simple and explains the result clearly. The key for initiating this path trace is the negative reduced cost of activity GUR, which is at its upper bound. (In another model, the bound could have been further up the path, rather than on the activity that yields the final product as its output.)

In the case of asking why not use coal, the rule finds the total delivered price, which is greater than the marginal price. Again, the key is the reduced cost because it reflects the difference between the activity’s total cost (direct + indirect), which is the delivered price, and the marginal price (converted to the activity’s units).
H.J. Greenberg / The ANALYZE rulebase

**Figure 8.** Electricity generation example.

**Figure 9.** Optimal tableau for electricity generation example.

**Figure 10.** A static explanation of electricity price.
ANALYZE ... INTERPRT PRICE ASK
What? demands electricity
There is 1 row:
DEL demands electricity.
What? how
By 3 fuels: coal, oil, and uranium.
What fuel? uranium
There are 2 activities associated with uranium:
PUR produces uranium with COST = $20.
GUR generates electricity from uranium with COST = $.40.
The total cost of these is $20.4, and the yield is .4. Thus, the
electricity price is $51. This is less than the marginal price
($52), but activity GUR is bounded by 10 units. The total
electricity generated from uranium is 4 units, which is less than the
total demand.
What fuel? coal
There are 2 activities associated with coal:
PCL produces coal with COST = $18.
GCL generates electricity from coal with COST = $.80.
The total cost of these is $18.80, and the yield is .33. Thus, the
electricity price is $56.97. This is greater than the marginal price
($52), so no coal is used.

Figure 11. Instantiating a prompt mode for price interpretation
(user entries are in italics).

Thus, in both cases the key to presenting an interpretation is understanding how
to explain a reduced cost. This is mathematically similar to explaining a dual price,
but rules of inference are different. Figure 11 shows these explanations, but let us go
further.

The first thing to notice about the dialog is that it is all in English, with no LP
jargon (except printing activity names, which could also be eliminated). The reasoning
behind the interpretation begins with the generation activities (positive entries in row DEL) and uses path tracing. The uranium price is less, but its path is blocked,
and the coal price is more.

The interpretation could have gone further into this from the vantage of explaining
the reduced costs. For example, the reduced cost of GCL is 1.64. This can be explained,
if the user asks directly for an interpretation of it, by the same breakdown
as above. The price difference of 4.97 (56.97 - 52) is in units of electricity. To put
this in units of coal, multiply by its yield: 1.64 = $4.97 \times 0.33.

An alternative explanation is to impute an electricity supply function, using dual
prices. Figure 12 shows an example of such a response (using one ANALYZE
command for the step function plot). The steps are determined by the same reasoning
as before, but the communication of the result is graphic. The step width is the total
amount of electricity that can be generated by the associated activity. The uranium
step is limited by the bound, GUR \leq 10; multiplying by the yield, this gives a
maximum of 4 units of electricity. Similarly, the oil step is limited by GOL ≤ 25; its
yield is 0.3, so it can generate up to 7.5 units of electricity. There is no coal bound,
so it can generate an infinite amount (indicated by the asterisk).

The step heights are the total imputed costs (that we computed before), and the
graphic view gives a bigger picture of the trade-offs. It also shows the range of
demand for which the character of the solution does not change: the basis remains
feasible, and hence optimal with the same prices, in the range of the oil step, which
is 4 to 11.5. (The fuels appear above their steps by automatic translation from the
syntax; it is an option of the ANALYZE command that was executed from within the
rule file.)

Let us consider one more example. A relevant portion of a coal model is pictured
in figure 13. There are 5 types of coal, distinguished by their sulfur and heat contents,
and each row is shown in full. The activities whose names are of the form Mijk mixes

Figure 13. A picture of a portion of an LP
with a qualitatively forced substructure.
coal types $i$ and $j$ to produce type $k$. The activities whose names are of the form $Oi$ operate plants that use coal type $i$ to generate electricity, and none of these plants operate in the solution. The user asks, Why not? Notice that the sign pattern is enough information to infer that all activities in this submatrix must have zero levels. The first row forces its adjacent activities to zero (because all activities are required to be non-negative). This eliminates the positive entries in row $L2$, which causes the remaining adjacent activities to be zero. The chain continues, successively forcing all of the activities to be at zero level.

The user might not quite understand the display shown in figure 13, but the same information is used to compose another response, like the one shown in figure 14. The query is answered omitting information about the other forced activities because they are not necessary to answer the query. The interpretation could therefore stop

```
ANALYZE ... INTERPRT WHY LEVEL O2
The level of O2 is 0. Here is the causal sequence of constraints that forces this.
Row L1 forces the levels of M132, M152, and M142 to be 0.
Row L2 forces the level of O2 to be 0.

Moreover, the levels of O1 and O3 are zero for the same reason: no activity supplies coal type 1, and this forces a sequence of activities (O and M classes) to be 0, including O1, O2 and O3. It is therefore not feasible in this LP for these activities to have a positive level.

If this is not a valid representation, the probable cause is inadvertent omission of activities that supply coal type 1.
```

Figure 14. A response to why a plant did not operate in the solution.

after the causal sequence, but the rule file anticipates that this could be a problem, so more information about related activities is given. Also, some advice is given about what the user might investigate (e.g., ask the model manager) if this result is not valid.

What we see emerging, in what might be considered a paradigm, is that the interpretation is in two parts. First, some portion of the LP is obtained that contains a causal interpretation, such as the submatrix that comprises a complete, margin-setting path to explain a price, or a forcing substructure that explains why some variables must be zero. Second, the isolated portion is explained with some mixture of English text, tables of numbers, and graphics. The first part is where most of the mathematical reasoning occurs, and the rules are algebraic or logical characterizations. The second part is called the discourse model in the system design, allowing the rulebase developer to choose the discourse that is most suitable for the model's constituency.
3.2 Dynamic/Whatif

Now consider rules for What if...? questions. These are dynamic in that the user wants to know how the solution would change in response to some data change or in response to forcing some activity to change its level. A key to the mathematical part of the reasoning is the rate of substitution between a basic and nonbasic activity. (In textbooks, these values are the negatives of the associated tableau entries, but ANALYZE shows tableau information as actual rates, rather than their negatives.) Although these also underlie why questions, we avoided using them in favor of path tracing in the interests of the final discourse. In a why question, the user is asking for a static interpretation that uses the original LP information, especially its structure. The rates, however, are imputed from the particular solution and have a dynamic quality.

To illustrate, consider the electricity generation example (figure 8). Figure 9 shows the full solution tableau, and we could look at the tableau column associated with row DEL to discover how the solution changes if electricity demand is changed.

Figure 15 shows the response to the question: What if electricity demand increases? It shows how activities change their levels, giving a cost breakdown that explains the $52 price. The rules for explaining the price, however, did not do this. In fact, it is not

```
ANALYZE ... INTERPRT WHATIF DEL INCREASES
Row DEL demands electricity, and its current value is 10. If this increases, the following activities change their levels at the rates shown.

<table>
<thead>
<tr>
<th>Basic</th>
<th>COST</th>
<th>Rate of Substitution (COST x Rate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>POL</td>
<td>15</td>
<td>3.333</td>
</tr>
<tr>
<td>GOL</td>
<td>.6</td>
<td>3.333</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>52</td>
</tr>
</tbody>
</table>
```

Figure 15. Instantiating a rule file to interpret the effect of increasing electricity demand.

wise in general to use the tableau rates if they can be avoided because it would tend to confuse a user who is not expert in LP. On the other hand, when asking for how the solution changes in response to a change in the demand, the user expects the response to be changes in levels and hence the total cost. (The rule concludes with a table that comes from one ANALYZE command, and activity names are shown. It is possible to replace these names with their meanings in English.)
In a simple model, like this example, the rates could be explained: rate of substitution = 1/(yield from oil) = 1/0.3 = 3.333. In most models, however, especially non-networks, such explanations are difficult because they involve basis inverse elements that need not be simply reciprocals.

Now consider a refinery example [5]. Figure 16 gives a flow schematic of processes that make two types of aviation fuels, called A and B. These are made from four blend stocks, and their qualities are measured by vapor pressure and octane number. The LP data is shown in the tables in figure 17.

![Refinery Blending Diagram](image)

(Figure 16. A refinery blending problem.)

<table>
<thead>
<tr>
<th>Product</th>
<th>Vapor Pressure</th>
<th>Octane Number</th>
<th>Demand (bbl/day)</th>
<th>Revenue ($/bbl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel A</td>
<td>7</td>
<td>90</td>
<td>1,300</td>
<td>6.50</td>
</tr>
<tr>
<td>Fuel B</td>
<td>7</td>
<td>100</td>
<td>800</td>
<td>7.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Blend Stock</th>
<th>Vapor Pressure</th>
<th>Octane at 1.2</th>
<th>Octane at 4</th>
<th>Supply (bbl/day)</th>
<th>Cost ($/bbl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alkylate</td>
<td>5</td>
<td>98</td>
<td>108</td>
<td>700</td>
<td>7.20</td>
</tr>
<tr>
<td>Cat. Cracked</td>
<td>6.5</td>
<td>87</td>
<td>94</td>
<td>600</td>
<td>4.35</td>
</tr>
<tr>
<td>Straight-run</td>
<td>4</td>
<td>80</td>
<td>87</td>
<td>900</td>
<td>3.80</td>
</tr>
<tr>
<td>Isopentane</td>
<td>18</td>
<td>100</td>
<td>108</td>
<td>500</td>
<td>4.30</td>
</tr>
</tbody>
</table>

(Figure 17. Data for refinery blending problem.)

The data is from Daellenbach and Bell [5], which gives the details of the formulation. Figure 18 shows the full LP. The demand equations for the two fuels are DA and DB, and the supply limits for the four blend stocks are SAL, SCC, SSR, and SIS. The quality constraints are $O_i$ and $V_i$ for the octane number and vapor pressure, respectively, for $i = A, B$. Figure 19 gives the optimal tableau, and the user asks what would
happen if we allocated catalytic cracked gasoline to make fuel B. This asks, *What if activity ACCB is forced to become positive?* (Numbers shown have been rounded to fit on this page.)

The rates are enough information to answer the query, but ANALYZE can perform pivots to see total effects (or to change the basis if we had a degeneracy that blocked this basis from responding to the perturbation). Figure 20 shows a response to this query. After obtaining the rates, the instantiated rule file translates the result into English sentences.

Figure 21 gives an alternative response, using English sentences. (This is an alternative format of the ANALYZE command executed, so the rule file does not have to compose the templates.) Other responses are possible. For example, it might be just the qualitative changes that are needed: which levels increase and which decrease. The reasons for the changes could, in principle, also be explained. For this example, that would not be difficult, but a generic rule to explain rates is difficult.
ANALYZE ... INTERPRT WHATIF ACCB INCREASES
Column ACCB allocates cat. cracked gasoline to aviation fuel B, and its current value is 0. If this increases, the following activities change their levels at the rates shown.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Cost Coefficient</th>
<th>Rate of Substitution (COST x Rate)</th>
<th>Net Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>AALA</td>
<td>7.2</td>
<td>.207</td>
<td>1.49</td>
</tr>
<tr>
<td>AALB</td>
<td>7.2</td>
<td>-.167</td>
<td>-1.2</td>
</tr>
<tr>
<td>ACCA</td>
<td>4.35</td>
<td>-1</td>
<td>-4.35</td>
</tr>
<tr>
<td>AISA</td>
<td>4.3</td>
<td>.164</td>
<td>.705</td>
</tr>
<tr>
<td>AISB</td>
<td>4.3</td>
<td>-.167</td>
<td>-.718</td>
</tr>
<tr>
<td>ASRA</td>
<td>3.8</td>
<td>.63</td>
<td>2.394</td>
</tr>
<tr>
<td>ASRB</td>
<td>3.8</td>
<td>-.667</td>
<td>-2.534</td>
</tr>
</tbody>
</table>

Total = -4.214

The total is for the indirect costs, which is offset by the COST of activity ACCB ($4.35), so the total rate of change in COST is $.135 ($ = 4.35 - 4.215). (Note: the possible inconsistency in the total is due to rounding the numbers before displaying them.)

Figure 20. A response to a What if query, using rates of substitution.

ANALYZE ... INTERPRT WHATIF ACCB INCREASES
A change in the level of COL that allocates cat. cracked gasoline to aviation fuel B affects the following.

+ .2068966 x COL that allocates alkylate to aviation fuel A
  (with COST = 7.199, so Net Rate = 1.489).
- .1666667 x COL that allocates isopentane to aviation fuel B
  (with COST = 4.3, so Net Rate = -.7166667).
+ .6293104 x COL that allocates straight run gasoline to aviation fuel A (with COST = 3.8, so Net Rate = 2.391).
- .6666667 x COL that allocates straight run gasoline to aviation fuel B (with COST = 3.8, so Net Rate = -2.533).
- 1 x COL that allocates cat. cracked gasoline to aviation fuel A (with COST = 4.349, so Net Rate = -4.349).
+ .1637931 x COL that allocates isopentane to aviation fuel A
  (with COST = 4.3, so Net Rate = .7043104).
- .1666667 x COL that allocates alkylate to aviation fuel B
  (with COST = 7.199, so Net Rate = -1.2).

...Total Net Rate = -4.214 = effect on COST from changes induced in the basic levels. Note that the reduced cost = COST Coefficient + Net Rate = 4.349 - 4.214 = .135345.

Figure 21. Another response to the What if query, using English translations.
In summary, many types of responses are possible, and this is part of the rulebase design, which is tailored to the particular constituency. The response should be static or dynamic according to whether the query is static or dynamic. Even though the same mathematics can be used for both why and what if questions, the user might not make the connection. Viewing dual prices as rates of objective change with respect to right-hand side perturbations is in the first place known only to someone knowledgeable in LP, and in the second place might not apply (owing to degeneracy). In the latter case, the character of the solution would have to change, such as changing the basis to find one that is compatible with the what if question.

4 Rules for debugging

4.1 Infeasibility diagnosis

Debugging means finding the cause of some LP having either an anomalous solution, or no solution. Here we consider only the latter, where the LP is either infeasible or unbounded. As in the case of sensitivity analysis, the intelligence breaks down into two parts: finding a causal substructure, and explaining the cause in problem-domain terms. Approaches to isolating a cause was given by Greenberg and Murphy [34], and particular ones used in ANALYZE have been given elsewhere [10, 12, 15, 16, 19, 21, 25] (see [32] for a mathematical foundation). An empirical study [18] that compares the quality of information from each of three primary isolation methods suggests that more than one is needed in the system, because no one method is completely dominate over the others.

One of the particular methods is to find an irreducible inconsistent subsystem (IIS), which has been studied extensively by Chinneck [3, 4]. MIMI/E [2] has a rule that keys off of the greatest phase 1 price (or reduced cost). The empirical study suggests this is a weak method, compared to the others, but ANALYZE has a command that uses these prices in a similar manner. In some instances, it provides a useful diagnostic, and its general advantage is that it requires less computation.

A rule file can use one method of isolation and try to use the information to diagnose the (probable) cause of the infeasibility. If it fails, either because the isolation is unsuitable or its interpretation is too difficult, it can try another method. Heuristics guide the choice of method, using whatever information about the LP it has. As before, simple, generic rules are tried first. One example is to look for a single row that cannot be satisfied, given the bounds on its adjacent activities. Perhaps surprisingly, this succeeds some of the time. The idea of a hierarchy of rules of increasing complexity was given in [15], where complexity pertains to the final explanation, rather than to the computations involved.

We shall illustrate three rules. The first is based on using Phase 1 dual prices ($\pi$), associated with the range constraints, $a \leq Ax \leq b$. First, form the aggregate range constraint, $\pi \alpha \leq \pi Ax \leq \pi \beta$, where
\[ \pi_i > 0 \Rightarrow \alpha_i = a_i \quad \text{and} \quad \beta_i = b_i; \]
\[ \pi_i < 0 \Rightarrow \alpha_i = b_i \quad \text{and} \quad \beta_i = a_i. \]

Second, compute the myopic range:
\[ \lambda = \operatorname{Min}\{\pi Ax : L \leq x \leq U\} \quad \text{and} \quad \mu = \operatorname{Max}\{\pi Ax : L \leq x \leq U\}. \]

It must necessarily be the case that either \( \mu < \pi \alpha \) or \( \lambda > \pi \beta \). For convenience, suppose \( L \geq 0 \). Then, the rule looks for nonzero coefficients, \( \pi A_j \), to address the question, \textit{Why is } \mu \textit{ so low?}, or \textit{Why is } \lambda \textit{ so high?}, in the respective cases.

In the case \( \mu < \pi \alpha \), we seek activities with capacity limits \( (U_j < \infty) \) or required positive levels \( (L_j > 0) \), according to whether the coefficient is positive \( (\pi A_j > 0) \) or negative \( (\pi A_j < 0) \), respectively. Binding non-negativity constraints \( (L_j = 0 \text{ and } \pi A_j > 0) \) are disregarded in the rule for obtaining a diagnosis because it is considered part of the model’s logic, whereas a binding capacity limit or positive requirement is a data element, which could be the cause.

To illustrate, consider the example in figure 22. This was described elsewhere [15, 26, 34], so we shall say only that it is a small network with no feasible flow. The above rule was applied, and the resulting interpretation was a reasonable diagnosis, except that the user might not recognize why the aggregate constraint is implied.

```
ANALYZE ... INTERPRT INFEAS
The constraints imply the following (aggregate) constraint:

\[ T25 + 2 \ T35 + T47 \geq 70 \]

where \( T_{ij} \) transports material from node \( i \) to node \( j \). These variables are bounded as follows:

\[ T25 \leq 10, \quad T35 \leq 10, \quad T47 \leq 2 \]

These bounds imply that the maximum of the left-hand side of the aggregate constraint is 32, so it cannot be satisfied.
```

Figure 22. An infeasibility interpretation for a network flow model using phase 1 price aggregation.

Figure 23 shows another interpretation using an IIS (computed from Chinneck’s code [4]), and figure 24 shows a graphic display formed by another rule that found a smaller subsystem.
ANALYZE ... INTERPRT INFEAS
The following portion of the LP is infeasible by itself:

\[ 0 = \text{NODE5} = T25 + T35 - T57 - T58 \]
\[ 20 \leq D7 = T47 + T57 \]
\[ 30 \leq D8 = T58 \]
\[ T25 \leq 10, \ T35 \leq 10, \ T47 \leq 2 \]

where Tij transports material from node i to node j, NODEi conserves flow through node i, and Dj requires demand at node j. It is necessary that some bound must change, or the equations must change, in order to become feasible.

Figure 23. An infeasibility interpretation for the network flow model using an IIS.

The following subnetwork has no feasible flow:

\[ 10 \]
\[ \boxed{10} \rightarrow \boxed{(5)} \rightarrow (8) \rightarrow 30 \]

Arrows are on arcs only where the direction of flow is part of the subsystem (this means the flow variable is non-negative, but if no arrow is shown, its non-negativity constraint is not part of the infeasibility).

Figure 24. A graphic infeasibility interpretation for the network flow model.

The next example illustrates two things. First, it shows how a rule can give a misleading diagnosis. Second, it illustrates another type of rule, based on successive reduction. Figure 25 gives an overview of a simple blending model. (The COST row is null because this was formulated to test rules that diagnose infeasibility.)

The infeasibility was formulated by adding row LPU to an otherwise feasible LP (using RANDMOD [15], which was designed for such testing purposes). This row requires total operation of all primary units to be at least 101, but the capacity rows (CAP) limit the total to 100. Figure 26 applies the same rule as in figure 22, but the result is misleading because it mentions only the operation of blend units, drawing no attention to the capacity limit rows.

The reason this happened is that the sum of infeasibility can be decreased by having more blend unit capacity, but removing all the blend unit capacity bounds is not enough. The real binding limit is the primary unit capacity constraints, but Phase 1 prices reflect marginal improvements, not the total change.
OP(PU,RF)  OB(BU)
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
LPU      1      >= 101
COST     ..MIN
:LO      0      0
:UP      *      20

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

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 finished product.
A row that begins with LPU limits total capacity of all primary units.
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.

Figure 25. Overview of a blending model.

ANALYZE ... INTERPRRT INFEAS
The constraints imply the following (aggregate) constraint:

1.272 OB03 + .5767 OB04 >= 65.147

where OB03 operates 3rd blend unit, and OB04 operates 4th blend unit.
These variables are bounded as follows:

OB03 <= 20, OB04 <= 20

These bounds imply that the maximum of the left-hand side of the aggregate constraint is 36.988, so it cannot be satisfied.

Figure 26. Instantiating the Phase 1 price aggregation rule to interpret the infeasibility of the blending model.
Figure 27. Another interpretation of the infeasibility in the blending model using successive bound reduction.

Figure 27 shows the interpretation from another rule, based on successive bound reduction. The blend units are included in the interpretation for the same reason as stated above, but at least the attention is drawn to the correct row (LPU) and its relation to primary unit capacities (CAP01, etc.). The table shown is produced by the ANALYZE command that performs successive bounding. The last paragraph is intended to address the question, How is row LPU related to the nonbasic activities shown? (Its original form is the sum of the OP activities.) The wording is perhaps cryptic, at least to one not expert in LP, and a better explanation could be composed from problem-domain information.

4.2 Unboundedness diagnosis

Now consider rules when the LP is unbounded. Mathematically, an unbounded LP means the dual is infeasible, so one approach is to apply rules for infeasibility diagnosis to the dual. Often, however, the cause is different. For example, figure 28 shows a variation of the electricity generation LP (figure 8), where there is a 2-step demand function. The first step sells electricity at $53 up to 10 units. If that were the only step, we would obtain the same solution as before, with a −$1 reduced cost on the SEL1 activity that would be at its upper bound (the $1 is the difference between the
Figure 28. An unbounded variation of the electricity generation LP.

revenue of $53 and the marginal price of $52). In this case, both steps, SEL1 and SEL2, are basic; the first is at its upper bound, and the second is at the end of the oil step (see figure 12). Unboundedness is detected with the nonbasic activity, GCL, which can generate more electricity at a net revenue of $1 (per unit of coal).

There are various ways to interpret the unboundedness to a user. If he/she is well acquainted with LP, we could identify the nonbasic activity with negative reduced cost and factor it. Figure 29 shows such a response.

Figure 29. An interpretation of the unbounded LP for a user with LP knowledge.

Variations of this response include replacing the basic variable names with their meanings. The table could also vary. Instead of using the original cost coefficient and imputed rates of substitution, the reduced cost factorization could be in terms of original matrix coefficients and dual prices. Figure 30 does this, showing how the total net rate of change in COST equals −1.
Figure 30. A variation of the interpretation in figure 29, using a different reduced cost factorization.

Figure 31. Another interpretation, using path tracing and English translations.

Figure 31 instantiates a different rule file, which uses path tracing to find an unblocked delivery path (it begins with the same nonbasic activity, GCL, and the electricity row, DEL). In this case, English sentences are given to the user. The last paragraph gives some advice about the probable causes, if the result is not valid.

Since each variation contains some different element of cognition for the user, one could have all varieties of responses in the rulebase. If the one chosen first does not provide enough help for the user to understand the cause, he/she can be prompted to ask for another interpretation. Then, another rule file can be instantiated to give a
different response. This could use the same information, varying only the communication, or it could seek another probable cause. Presently, the rules for choosing one interpretation over another is an art, but one can go further by asking or inferring information about the user's skill, particularly in LP. Internally, a skill category can be set, initially to some default for the constituency. As the session progresses, this skill category can change, giving a different priority to which rule file gets instantiated for subsequent queries.

5 Rules for model management

In this section, I show how ANALYZE can provide support for several model management functions. Besides its ongoing use, one class of related functions is documentation, verification, and validation. Another function of model management is periodic review in order to simplify the model. These functions will be considered first. Then, I shall describe how to test new rule files, as the need arises for ongoing analysis support.

5.1 Documentation, verification, validation

Some documentation aids have already been illustrated, such as the overview given in figure 2. Mixtures of English text, tables of numbers and graphic displays were described elsewhere [28], and these can be extended to give reports across scenarios using MODLER [23]. The reports can apply reasoning about the results, rather than just state what they are.

Verification is performed by preparing some key elements to check, then retrieving what is in the LP and comparing them. The elements could be simply data values, or they could also be structural properties. For example, figure 32 shows an embedded cycle in an LP matrix, where the columns are shown in full. Activity RD

\[
\begin{array}{ccc}
\text{RD} & \text{DG} & \text{GR} \\
\text{COST} & \sigma_R & \sigma_D & \sigma_G \\
\text{RES} & -1 & \gamma_R & \geq 0 \\
\text{DIS} & \gamma_R & -1 & \geq 0 \\
\text{GAS} & \gamma_D & -1 & \geq 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{RD} & \text{DG} & \text{GR} \\
\text{COST} & \sigma_R / \gamma_R & \sigma_D / \gamma_D & \sigma_G & \geq 0 \\
\text{RES} & -1 & \gamma_R & \gamma_D & \gamma_G & \geq 0 \\
\text{DIS} & 1 & -1 & \geq 0 \\
\text{GAS} & 1 & -1 & \geq 0 \\
\end{array}
\]

(a) Original Form

(b) Scaled Form

Figure 32. An embedded cycle.

converts residual oil to distillate oil, and DG converts distillate to gasoline. These are physical processes that can be done in a refinery with yields \( \gamma_R, \gamma_D > 1 > \gamma_G \) (converting from a heavier to a lighter petroleum product increases the volume). The last
activity, GR, converts gasoline to residual oil, but it does not represent a physical process. Its presence in the LP is to bound relative prices to make the results better match reality that might be compromised by the linearity assumptions.

Putting aside further discussion about the need for this cycle, especially activity GR, which is an artifact of the overall modeling, consider an implication of its presence. The net throughput is the product of the yields: \( \tau = y_R y_D y_G \). One unit of RD results in \( \tau \) units of RD at the end of the cycle. This is evident in the scaled form, which was obtained by first dividing column RD by \( y_R y_D \) and column DG by \( y_D \), then multiplying row RES by \( y_R y_D \) and row DIS by \( y_D \).

If the yields were calculated correctly, \( \tau = 1 \), and this could be verified. For any \( \tau \geq 1 \), the net cost around the cycle is the sum: \( c_R/y_R y_D + c_D/y_D + c_G \). Suppose there are no capacity limits on these three activities. Then, for the LP to be bounded, it is necessary that the net cost be non-negative. An element of verification is to check that this holds (strictly positive to avoid unboundedness due to numerical error). An intelligent aid to verification is to detect such cycles, and automatically write the test to a rule file. The model manager can instantiate this rule file whenever the particular costs or yields change.

### 5.2 Model simplification

Now consider model simplification, which is done periodically as part of good model management. One form is dimensional reduction. Applied to just one scenario, not much is discovered for general model reduction. An exception might be qualitative inferences, such as the forcing structure shown in figure 13, if it is generally the case, rather than particular to the instance. (It is always important to distinguish whether some variables are forced by economic trade-off or by implication of the constraints.) Removing variables and constraints that are always forced is a part of good model management.

By testing reductions for a significant sample of scenarios that were run since the last review (say about 3–6 months), a rule file can be constructed to identify redundancies and other indications of potential reductions. These can be organized into a summary report with advice given to the model manager for reductions that matter.

What matters is not just the time it takes to solve an instance, but also how extraneous rows and columns can obscure the documentation and unnecessarily complicate analysis, particularly with heuristics that might get misled into some dead ends. One example of an important reduction is described in [27], which reduced the LP to triviality.

Another form of model simplification is structural [33]. A young model might have many features that the formulator thought would be useful for certain scenarios, giving analysts flexibility to increase the scope of the model over time. After some use, these features might actually obscure the simplicity of the model’s structure. For
example, one might have an equivalent network formulation, which is much simpler to document and manage. Automatic reformulation has received some attention over the past 15 years, and most methods seek particular substructures. These heuristics can be put into a rule file, using ANALYZE procedures that seek embedded structures.

Validation pertains to how well the model represents the system, at least in the context of the quality of decision support. One example of a validation exercise is to compare market prices with LP prices. This can be done using the price interpretation rules I presented earlier, except now the purpose is different. More generally, the methods of post-optimal sensitivity analysis can apply directly to a family of validation exercises, except the final interpretation could be different. Further, a database could be read that contains prices, or other data, against which to compare the LP results. Significant deviations can be compared either informally, or with a statistical package.

5.3 Developing and testing new rules

The last model management function I shall consider is testing new rule files. It can happen that a model manager has several different rules in mind for the same interpretation, and he/she wants to test them with different cases to see how each rule file variation behaves. One way to do this is to change the keyworded file, RULBASE.RUL (see appendix), to have the name of each rule file in the list. Another way is to have a single option, which I shall call TEST, which instantiates the particular rule file being tested.

To illustrate, we have a forestry model, and we want to show the distribution of harvesting that meets demands in each of several time periods. Gathering the information is simple, but the issue is communication.

Figure 33 shows the first variation. The rule file TEST is passed the name of the rule file under consideration, which is DISTRIB1. All that TEST does is check a few things (common to all the rules) and instantiate the test rule file. In this variation, a block picture is presented to the user, followed by its meaning.

Figure 34 instantiates the second variation by passing DISTRIB2 as the parameter to TEST in the INTERPRT specification. This variation is more succinct and does not present a block picture to the user. It simply goes through each area for which harvesting occurs in some time period, and gives a sentence that says which time periods each area has positive levels of harvesting.

Figure 35 shows the third variation. The response is a listing of the demand equations with the harvest activities having positive level. Following the equation listing, each harvest activity is explained.

These three rule files can be applied to a variety of scenarios, and results shown to a sample of users for feedback. One of these (or a new one) enters the production system by putting its name into the keyworded file, RULBASE.RUL.
Figure 33. Instantiating the first distribution rule.

Figure 34. Instantiating the second distribution rule.

6 Summary and conclusions

The examples given here demonstrate that it is possible to design rules that not only give interpretations that are comparable to what an LP expert would give, but also that the results can be communicated effectively. The ANALYZE system is designed to support rule-based interpretations, which create an artificially intelligent environment for LP analysis support, and a multi-view architecture that gives flexibility in composing the interpretation.
Figure 35. Instantiating the third distribution rule.

Although the examples were small, the underlying concepts extend to realistic sizes. When facing a very large LP, say tens of thousands of equations and hundreds of thousands of variables, the relevant portion of the LP can also be too large to display. In that case, there are condensation methods (such as blocking), which ANALYZE can perform to visualize what is important. For example, suppose we isolate an infeasibility to consist of two regional process models (like refining and electricity generation) plus some transportation links. The embedded process models can be condensed into one node each, such as by the syntax of the LP, making it easy to see a graph with two nodes and one link. Zooming operations [35] can then be used to see some details within the process models.

The three functional categories given here are not exhaustive, and other rule files provide additional support. Most of this has been with basic solutions because that is the history of LP. Now that interior point methods have become viable alternatives, new procedures are needed to provide causal information from a strictly complementary solution. Examples of when an interior solution provides better information than a basic solution have been indicated by several authors [29,36,38,39], and new research is in progress to extend those results. This is one avenue for further research: how to obtain algebraic or logical substructures that provide causal information, even if the solution is not basic.

Another avenue for further research is in the discourse models. The English responses illustrated here use the LP syntax [11,28], which is part of the model formulation. (MODLER and GAMS can automatically generate the syntax file, which ANALYZE reads to obtain the LP syntax.) An analogical approach is with neural networks [17], not yet fully developed. Jones [37] gives several avenues to pursue, including possible uses of animation and virtual reality.
Acknowledgements

This research was partly supported by Chesapeake Decision Sciences, Hewlett Packard, IBM, Primal Solutions, and Shell Development Company. Partial support was also provided by the Energy Information Administration.

Appendix

ANALYZE is implemented in FORTRAN/77 and runs in many computing environments, including DOS, Unix, VMS, CMS, and TSO. This appendix gives some of its particulars, including elements of a rule file. Further information about the capabilities is given in the User’s Guide [22].

Figure 36 lists most of the ANALYZE commands in a menu format, showing the types of functions they perform. This gives an overview with only one screen. More information about each command, and what it does, is also available as a quick reference, and the HELP command gives more information and examples about each command. There are also extensive document files and examples to which the user is referred for further guidance.

<table>
<thead>
<tr>
<th>Control</th>
<th>I/O</th>
<th>Delineate</th>
<th>Edit</th>
<th>Sensitivity</th>
<th>Assistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUIT</td>
<td>EXECUTE</td>
<td>BLOCK</td>
<td>FIX</td>
<td>AGGREGAT</td>
<td>HELP</td>
</tr>
<tr>
<td>RECAP</td>
<td>OUTPUT</td>
<td>SUBMAT</td>
<td>FREE</td>
<td>BASIS</td>
<td>INTERPRT</td>
</tr>
<tr>
<td>SCREEN</td>
<td>PRINT</td>
<td>SUBMAT</td>
<td>RENAME</td>
<td>RATEOF</td>
<td>?</td>
</tr>
<tr>
<td>STRING</td>
<td>READIN</td>
<td>ROUND</td>
<td>REDUCE</td>
<td>TRACE</td>
<td></td>
</tr>
<tr>
<td>SWITCH</td>
<td>SOLUTION</td>
<td>WRITEOUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_DICTION</td>
<td>WRITEOUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_SETUP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query

<table>
<thead>
<tr>
<th>ROW or COL</th>
<th>ROW and COL</th>
<th>Syntax</th>
<th>Special</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRIM</td>
<td>COUNT</td>
<td>EXPLAIN</td>
<td>GRAPH</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>LIST</td>
<td>SCHEMA</td>
<td>SUMMARY</td>
</tr>
<tr>
<td>SHOW</td>
<td>PICTURE</td>
<td>SYNTAX</td>
<td></td>
</tr>
<tr>
<td>TALLY</td>
<td>SHOW //ARRAY</td>
<td>TABLE</td>
<td></td>
</tr>
</tbody>
</table>

Figure 36. Menu of ANALYZE commands.

The remainder of this appendix gives particulars about rule files. Figures 37–40 list the keywords, which can be referenced in the text for simple lookup. The character, integer, real and logical rule keys are listed separately with a brief description of what they mean. In the case of a character keyword, substrings can be referenced. For example, %OPT(1:3) becomes MIN or MAX in the translation. The logical keywords are switches whose value is either T (true) or F (false).
### Figure 37. Character rule keys.

<table>
<thead>
<tr>
<th>Key</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOUND</td>
<td>bound set name</td>
</tr>
<tr>
<td>COLST</td>
<td>current column status</td>
</tr>
<tr>
<td>COLUMN</td>
<td>current column name</td>
</tr>
<tr>
<td>OBJ</td>
<td>name of objective row</td>
</tr>
<tr>
<td>OPT</td>
<td>sense of optimization (MINIMIZE or MAXIMIZE)</td>
</tr>
<tr>
<td>PROBLEM</td>
<td>problem name</td>
</tr>
<tr>
<td>RANGE</td>
<td>range set name</td>
</tr>
<tr>
<td>RHS</td>
<td>right-hand side set name</td>
</tr>
<tr>
<td>ROW</td>
<td>current row name</td>
</tr>
<tr>
<td>ROWST</td>
<td>current row status</td>
</tr>
<tr>
<td>STATUS</td>
<td>solution status (overall)</td>
</tr>
<tr>
<td>_NAME(1)</td>
<td>name of i-th vector entry</td>
</tr>
</tbody>
</table>

### Figure 38. Integer rule keys.

<table>
<thead>
<tr>
<th>Key</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAMELN</td>
<td>length of row and column names</td>
</tr>
<tr>
<td>NCBLKS</td>
<td>number of column blocks</td>
</tr>
<tr>
<td>NCOLS</td>
<td>number of columns in submatrix</td>
</tr>
<tr>
<td>NCSYN</td>
<td>number of column classes in syntax</td>
</tr>
<tr>
<td>NESYN</td>
<td>number of entity sets in syntax</td>
</tr>
<tr>
<td>NONES</td>
<td>number of ones in submatrix</td>
</tr>
<tr>
<td>NONZERO</td>
<td>number of nonzeros in submatrix</td>
</tr>
<tr>
<td>NRBLS</td>
<td>number of row blocks</td>
</tr>
<tr>
<td>NROWS</td>
<td>number of rows in submatrix</td>
</tr>
<tr>
<td>NRSYN</td>
<td>number of row classes in syntax</td>
</tr>
<tr>
<td>NSTACK</td>
<td>max number allowed in stacks 1 and 2</td>
</tr>
<tr>
<td>NSTACK1</td>
<td>number in stack 1</td>
</tr>
<tr>
<td>NSTACK2</td>
<td>number in stack 2</td>
</tr>
<tr>
<td>NTAB(1)</td>
<td>i-th tab setting (for i = 1 to 10)</td>
</tr>
<tr>
<td>NVALUES</td>
<td>number of distinct nonzeros</td>
</tr>
<tr>
<td>NVECTOR</td>
<td>number of entries in VECTOR and _NAME</td>
</tr>
<tr>
<td>NZCOL</td>
<td>number of nonzeros in column</td>
</tr>
<tr>
<td>NZCUB</td>
<td>number of nonzeros in column for submatrix rows</td>
</tr>
<tr>
<td>NZROW</td>
<td>number of nonzeros in row</td>
</tr>
<tr>
<td>NZRSUB</td>
<td>number of nonzeros in row for submatrix columns</td>
</tr>
<tr>
<td>Key</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>VCOLC</td>
<td>Objective value of current column</td>
</tr>
<tr>
<td>VCOLD</td>
<td>Reduced cost of current column</td>
</tr>
<tr>
<td>VCOLLO</td>
<td>Lower bound of current column</td>
</tr>
<tr>
<td>VCOLUP</td>
<td>Upper bound of current column</td>
</tr>
<tr>
<td>VCOLX</td>
<td>Level of current column</td>
</tr>
<tr>
<td>VDENS</td>
<td>Density of LP matrix</td>
</tr>
<tr>
<td>VECTOR(i)</td>
<td>i-th vector value</td>
</tr>
<tr>
<td>VLOOK</td>
<td>Value from last LOOKUP</td>
</tr>
<tr>
<td>VRWC</td>
<td>Objective value of current row (= 0, except objective)</td>
</tr>
<tr>
<td>VRWLO</td>
<td>Lower bound of current row</td>
</tr>
<tr>
<td>VRWP</td>
<td>Price of current row</td>
</tr>
<tr>
<td>VRWUP</td>
<td>Upper bound of current row</td>
</tr>
<tr>
<td>VRWY</td>
<td>Level of current row</td>
</tr>
</tbody>
</table>

Figure 39. Real rule keys.

<table>
<thead>
<tr>
<th>Key</th>
<th>Meaning if value = T (true)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWMSG</td>
<td>Message switch is on</td>
</tr>
<tr>
<td>SWOTFIL</td>
<td>Output is going to a file</td>
</tr>
<tr>
<td>SWINFIL</td>
<td>Input is coming from a file</td>
</tr>
<tr>
<td>SWRATE</td>
<td>Basis is setup (to support RATEOF command)</td>
</tr>
<tr>
<td>SWYSYN</td>
<td>Syntax has been read in</td>
</tr>
</tbody>
</table>

Figure 40. Logical rule keys.

Figure 41 lists the commands recognized from a rule file. The first executes any ANALYZE command (except INTERPRT, which is listed separately to avoid recursion since re-entrance into a subroutine is disallowed by the standard FORTRAN). The other commands are particular to rule files, and they include branching (possibly conditional) and looping to control the logic based on the LP information obtained.

The keyworded file, RULBASE.RUL, is interrogated before the interactive session begins. This determines the list of INTERPRT options that the user sees. A rule file, however, can instantiate other rule files not listed in RULBASE.RUL. This is useful for transferring to more specialized rules, perhaps exploiting the particular LP model or model class. It is also useful for testing new rules.
<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYZE</td>
<td>execute an ANALYZE command</td>
</tr>
<tr>
<td>ASK</td>
<td>prompt user (from within rule file)</td>
</tr>
<tr>
<td>CALC</td>
<td>perform simple arithmetic on a parameter</td>
</tr>
<tr>
<td>DEBUG</td>
<td>interrupt the rule file and enter debug mode</td>
</tr>
<tr>
<td>ENDLOOP</td>
<td>end a loop</td>
</tr>
<tr>
<td>ENTITY</td>
<td>retrieve the meaning of a set member</td>
</tr>
<tr>
<td>EXIT</td>
<td>terminate the rule file</td>
</tr>
<tr>
<td>FIND</td>
<td>find a syntax element whose meaning matches a specified string</td>
</tr>
<tr>
<td>FORM</td>
<td>form a row or column class set member</td>
</tr>
<tr>
<td>GOTO</td>
<td>branch to a label or to the top of the rule file</td>
</tr>
<tr>
<td>IF ... THEN</td>
<td>conditionally execute a command</td>
</tr>
<tr>
<td>INTERPRT</td>
<td>instantiate another rule file</td>
</tr>
<tr>
<td>LOOKUP</td>
<td>get an LP value</td>
</tr>
<tr>
<td>LOOP</td>
<td>begin a loop (e.g., over submatrix rows or columns)</td>
</tr>
<tr>
<td>NEXT</td>
<td>increment a loop (e.g., next row or column)</td>
</tr>
<tr>
<td>POP</td>
<td>retrieve the top item in a stack</td>
</tr>
<tr>
<td>PUSH</td>
<td>store an item onto the top of a stack</td>
</tr>
<tr>
<td>QUEUE</td>
<td>store an item onto the bottom of a stack</td>
</tr>
<tr>
<td>SET</td>
<td>set the value of a parameter</td>
</tr>
<tr>
<td>SKIP</td>
<td>skip lines or to the end of a loop</td>
</tr>
<tr>
<td>STACK</td>
<td>initialize stack(s)</td>
</tr>
<tr>
<td>TEXT</td>
<td>set formatting of text</td>
</tr>
<tr>
<td>VECTOR</td>
<td>define or edit a vector (could be set by a procedure, like rates)</td>
</tr>
</tbody>
</table>

Figure 41. Rule file commands.

References


